51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2011年7月
总第004期

HTML 5开发
没有捷径

掀起C++ 11
神秘面纱

谷歌史上最大失误
放弃Java

developer.51cto.com

目录 2011.07 Contents

51CTO.com 技术成就梦想

出版方：北京无忧创想信息技术有限公司　责任编辑：彭凡

编辑：周少南　陈贻新　边艾婧　封面设计：徐泽琼

《开发月刊》欢迎大家踊跃投稿，来稿请注明“开发月刊投稿”并发送至 pengfan@51cto.com

出版日期：2011 年 7 月 14 日（每月 14 日出版）

7月编程榜：COBOL五角大楼出品

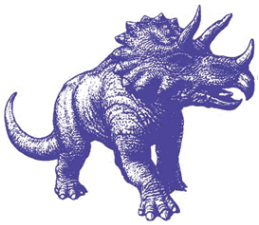
7 月 10 日，Tiobe 发布了新一期编程语言排行榜。2011 年已经过去一半，这期 Tiobe 的主题为今年的“年度语言”预测。Lua 和 Objective-C 成为 2011 年“年度语言”热门人选。而本期的 51CTO 编程语言榜将向大家继续介绍古典语言，本期主题为“COBOL 五角大楼出品”。

参考今年的语言发展走势以及市场占有率，Tiobe 的预测目标为今年比较火爆的移动开发语言：Objective-C 和 Lua。其中 Objective-C 相对 2010 年增长了 2.68%，亚军 Lua 增长率为 1.04%，如果按这个走势下去，2012 年的 1 月，Lua 也将和现在的 Objective-C 一样成为年度语言热门人选。前三甲虽然是 Java、C、C++，但是从下图的榜单中可以看出 Objective-C 和 Lua 借苹果之势，使用的人也越来越多了。

Position Jul 2011	Position Jul 2010	Delta in Position	Programming Language	Ratings Jul 2011	Delta Jul 2010
1	1	=	Java	19.251%	+0.58%
2	2	=	C	17.280%	-1.20%
3	3	=	C++	9.017%	-1.45%
4	5	↑	C#	6.221%	+0.49%
5	4	↓	PHP	6.179%	-2.39%
6	9	↑↑↑	Objective-C	5.181%	+2.68%
7	6	↓	(Visual) Basic	5.106%	-0.41%
8	7	↓	Python	3.583%	-0.63%
9	8	↓	Perl	2.328%	-0.77%
10	10	=	JavaScript	2.242%	-0.19%
11	19	↑↑↑↑↑↑↑↑	Lua	1.572%	+1.04%
12	12	=	Ruby	1.325%	-0.66%
13	16	↑↑↑	Lisp	0.906%	+0.28%
14	11	↓↓↓	Delphi/Object Pascal	0.887%	-1.44%
15	24	↑↑↑↑↑↑↑↑↑	Transact-SQL	0.802%	+0.34%
16	15	↓	Pascal	0.668%	+0.03%
17	-	=	Assembly*	0.618%	-
18	22	↑↑↑↑	RPG (OS/400)	0.559%	+0.09%
19	28	↑↑↑↑↑↑↑↑	Ada	0.549%	+0.17%
20	46	↑↑↑↑↑↑↑↑↑↑	C shell	0.545%	+0.33%

上图是前 20 名编程语言排行：

今天 51CTO 的编程语言排行榜我们要介绍的这款编程语言，相信很多中国程序员都使用并开发过。早在上世纪 50 年代 COBOL 语言就已经投入使用，并渗透到商业软件领域。据统计，目前有 1000 亿行 COBOL 代码被开发出来，并有日益红火的趋势。不少中国对日外包就在使用 COBOL 进行开发。



COBOL 于 1959 年 5 月，五角大楼委托格雷斯霍波（G.Hopper）博士领导一个委员会并由 RearAdmiralGraceHopper 公司主持开发。最开始的目的就是用于海量数据信息的处理，虽说五角大楼宣称 COBOL 是用于企业管理等商业领域，但这种大量数据处理恐怕也是为了军事目的服务，如导弹弹道计算、情报数据综合分析等领域。

对于大量数据进行快速处理的需求，催生了 COBOL 这样擅长数据项和输入、输出记录处理，对大量数据文件提供简单处理方式的语言。

1959 年，美国国防部召开了一个有政府机关

6 月编程语言排行榜：COBOL 五角大楼出品 II

企业、计算机厂商参加的会议，大家认为有必要设计一种数据处理专用的语言，这就是著名的数据系统语言会议 CODASYL（Congferenceon DataSystemsLanguages）。1959 年 12 月出现了第一个 COBOL 语言文本，并于 1960 年 4 月正式发表，因此也被称为 COBOL-60。

2009 年，COBOL 进入了第五十个年头。

COBOL 依旧红火

40 年过去了，编程语言排行榜上 COBOL 早已不是最受关注语言。比它晚 30 多年的 Java 和 C 语言正为了榜首争得不可开交。但是在财会工作、统计报表、计划编制、情报检索、人事管理等数据管理及商业数据处理领域，COBOL 依旧红火。

COBOL 掌控全球的钱袋子？

COBOL 主要应用在银行等金融领域，即使现在他们想换到其他类型的语言。巨大的迁移成本和风险将让他们望而却步，况且现在 COBOL 还工作得很理想，为什么要更换呢？

COBOL 的重要性可以用这句话来描述：世界上 70% 的数据都是用 COBOL 语言处理的，并且 90% 的 ATM 事务处理用的都是 COBOL 语言。每天在线处理的 COBOL 事务有 300 亿次。500 强中有 492 家（包括全部的 100 强）使用了 COBOL 语言，目前在 COBOL 方面的投资已经超过 3 万亿美元。

COBOL 语言的 Hallo World 代码

```
******************************
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
MAIN SECTION.
DISPLAY "Hello World!"
STOP RUN.
******************************
```

从上面的 COBOL 代码，我们可以看出 COBOL 程序由 4 部 (DIVISION) 组成：

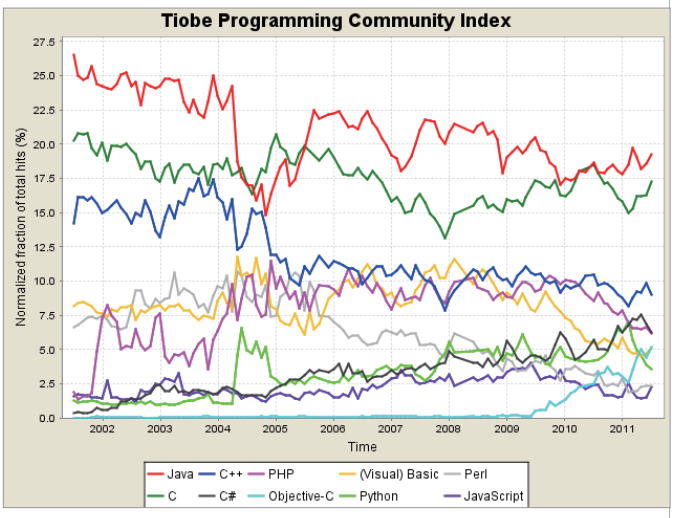
IDENTIFICATION DIVISION.（标识部），主要用来指定源程序名字，也可以写入其他用作备忘的某些信息（如日期、作者等）。

ENVIRONMENT DIVISION.（环境部），主要用于指出程序中用到的数据文件名与计算机系统的设备的对应关系，即把某一文件名与一个外部设备联系起来。（未完内容请看网络原文：

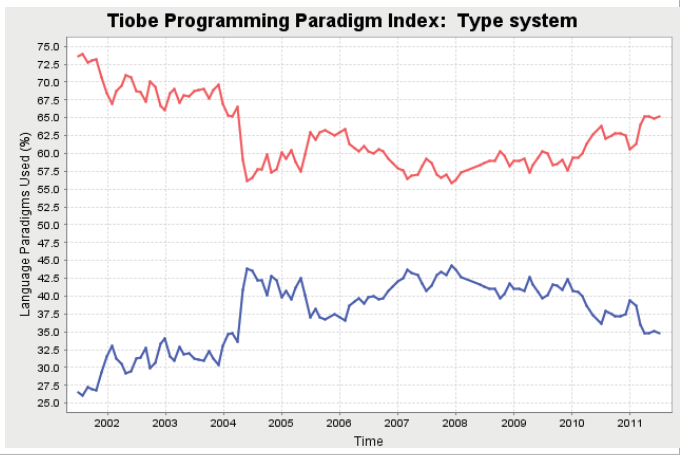
http://developer.51cto.com/art/201107/273497.htm

下面是本期编程语言排行榜的其他排名数据和趋势走向。■

前 10 名编程语言走势图



下面给出了编程语言类别的一年变化趋势



HTML 5 是近十年来 Web 开发标准最巨大的飞跃。和以前的版本不同，HTML 5 并非仅仅用来表示 Web 内容，它的新使命是将 Web 带入一个成熟的应用平台，在 HTML 5 平台上，视频，音频，图象，动画，以及同电脑的交互都被标准化。

专题：HTML 5 下一代 Web 开发标准详解

http://developer.51cto.com/art/200907/133407.htm

专题：HTML 5 来袭：WEB 前端开发面临十字路口

http://developer.51cto.com/art/201106/269101.htm

究竟 HTML 5 是什么样子？我们将带您走进 HTML 5 的世界。

51CTO 开发频道寄语

HTML 5在应用程序开发方面没捷径

核子可乐译

对于专业开发人员来说，最令人抓狂的说法莫过于有人以“HTML 程序员”自居。网页编码之于真正的编程工作，正如撰写菜单之于真正的烹饪过程。但平台供应商近来的说法无疑令人大跌眼镜——HTML 已成为从智能手机、平板应用到桌面系统应用等各大开发领域首选开发工具。

Palm 品牌当初启动其自有平台 WebOS 时，曾声称开发者只需掌握基本的网页编程标准，即可轻松为其开发应用程序。微软在 Windows Phone 7 上也放出过类似言论。谷歌 Chrome 浏览器具备一套“网络商店”，允许大家将自己的桌面网页应用程序放入其中进行出售。但最终真正令我瞠目结舌的情况是，根据最近演示版本的表现，Windows 8 系统上的应用程序将主要利用 HTML5 进行开发。这令 Windows 开发人员们惊恐不已之余，更令人不禁揣测微软可能正计划完全放弃对 Silverlight 甚至 .Net 本身的支持。

难道我们真会被HTML 5铺天盖地的宣传攻势所蒙蔽，继而相信微软已经准备放弃 Windows API 并转而支持网页标准？这毫无道理。我承认 HTML5 的确是一款不错的工具，而且能够为网页创造出良好的应用，但如今将其推至如此的高度就太过荒谬了。尽管它广受欢迎，但我们仍然有足够的理由解释为什么没人将其作为通用开发工具的首选方案。以下是需要深入思考的一些注意事项。

1. 打算用 HTML 单独创建程序？

任何建议用 HTML 来创建应用程序的家伙实际上都打定了主意想拖我们的后腿。他们潜在的台词当然是让我们用 HTML 及 JavaScript 来编写程序，但这还不是我们需要用到的全部工具。事实上，要开发真正的网页应用程序，我们至少要用到 HTML、JavaScript 以及 CSS——三种各自完全独立的语言，而且是同时用到。W3C 曾努力为 HTML5 添加了更多 API，以使其能够与网页标准及诸如多线程及本地存储之类的功能共同协作。这种做法其实是先假设我们的应用程序不再连入任何类型的服务器端组件——也许是为了减轻繁重的运算及存储负荷——而接下来我们要面对的就是所有必要的额外语言、API 以及标准。

当有人谈起创建某种应用程序“如同创建网页应用程序一样简单”时，他们所表达的真正意义是什么？网页开发当下已经演变出一套复杂、多层次且涉及多种语言的规则。一般来说开发网页应用已经不像我们印象中那么小菜一碟了。但这种难于掌握的模式真的就是我们打算强加给下一代开发人员的传承产物吗？

2. HTML 的设计初衷并非为创建应用程序

HTML5 总是伴随着许多议论，最主要的论调是认为它本身就是 HTML 的强化版，只不过针对网页应用程序支持做出了一些改进。但拥有更好的应用程序支持并不代表着 HTML 标准也在向此方向转型。原本从 XHTML 1.1 手中接过旗帜的应该是 XHTML 2，其着重强调的是语义标识与 XML 集成。归根结底，XHTML 2 是一种以文档为核心标识的语言。

HTML 5 在应用程序开发方面没有捷径 II

XHTML 2 最终折戟沉沙，然而，一个名为网页超文本应用程序技术工作组（简称WHATWG）的独立机构却从该项目中分裂出来，不仅摆脱了 W3C 的 HTML 构想，更开始从另一个角度为新标准的诞生努力工作。而正是由于该机构的出现，网页应用程序才迎来了崭新的纪元——其研究的成果正是我们今天所熟知的 HTML 5 的基础。

但是 HTML 5 真的就是最好的发展方向吗？以 HTML 5 所大肆宣扬的 <canvas> 标签为例，其从本质上意味着“插入一系列编程生成的、无法由标识描述的图形内容”。这对于标识语言来说是种很奇怪的使用方式。如果我们沿着这种思路继续走下去，恐怕等于是将网页标准硬塞进一个它永远不会真正适应的躯壳当中。这对于网页领域来说可能是必要的，但我们真的希望将这套体系扩展至方方面面，并最终把自己也扯进极为被动的局面中？

3. HTML 在构建用户界面方面弱爆了

苹果公司对其原始 Mac 电脑做出的最重大的革新措施之一就是为开发者们公布了一套细致的人机接口指南。因此，与 DOS 程序不同，Mac 电脑上的应用程序在外观和实际表现方面具备良好的一致性。这些应用全都采用同样的菜单、同样的对话框以及同样的报错方式。当大家探讨 Mac OS 的巨大成功时，该系统在印象上的连贯性与一致性被公认是其中一个突出的原因。这种特性甚至能克服一部分新系统给用户带来的陌生感与困惑。

而在网页应用程序方面，我们又回到了 DOS 时代。界面设计者们能够自由地创建他们想要的任何按钮，可以让菜单从任何位置滑下或弹出，也完全有能力按自己的喜好将整个窗口设置成任何样式。如果没有一套标准化的部件，利用网页技术开发应用程序会使成品变得充满冲突感，有时甚至会让人感觉像是外星人的作品。此外，即使有些人愿意彻底抛开自己的想法，将用户界面构建得与 iPhone 上的应用程序一模一样，这套界面框架仍然无法直接适用于其它类型的 Android 手机。谁会花费大把时间去创建那种一个平台一个样的“本地化”网页应用？绝对没人会这么做。

4. 创建专有平台 HTML 应用程序无意义

好吧，退一步说，也许大家并不在意具体为哪款设备或哪个平台开发软件。比如说，现在我们正为 iOS 或 Windows 8 开发一款应用程序。但我们究竟为什么要在单平台应用程序开发中使用 HTML 呢？事实上 HTML 的全部卖点及其相关技术所指向的都是开放式、跨平台标准。

更重要的是，iOS 与 Windows 都具备自己的软件开发工具，而且这些工具在许多 HTML 有所不足的方面还同时做出了强化。首先它们为大家提供了一套标准化部件，允许我们建立统一的用户界面。其次，它们还提供访问 API 的权限，使我们能够根据本地处理速度对运算强度进行调整。此外，它们允许大家将自己的应用程序与那些其它平台不具备的核心 OS 特色整合起来（由此也能够大概推测消费者为什么会选择此类平台）。而如今，我们要把这种种优势全数抛弃，为什么？ 未完内容请看原文：

http://developer.51cto.com/art/201107/275248.htm ■

有关HTML 5的流言与真相

作者 /neodreamer

你是免不了的。每个人都在谈论 HTML 5。自众人开始滥用圆角和渐变效果以来，HTML5 或许是最热炒的技术。然而，许多人眼中所谓的 HTML5 实际上只是老式的 DHTML 和 Ajax。有关 HTML5 的诸多信息中鱼目混珠，因此 ,JavaScript 专家雷米 · 夏普 (Remy Sharp) 和 Opera 公司的布鲁斯 · 劳森 (Bruce Lawson) 着眼这些流言，对其中常见谬误和事实做了分类整理。

首先，一些事实

很久很久以前，世上有一门叫做 HTML 的可爱语言，这门语言简单易学，用它写网站真是轻而易举。因而，所有人都用这门语言，从此，Web 也从一堆物理论文的链接变成了今天我们所熟知和喜爱的模样。

大多数页面并不遵循这门语言的简单规则 (因为写这些网页的人对内容本身要比媒介形式更为关心)，因此所有浏览器都必须忽略错的代码，尽最大努力猜测作者到底是想怎样展示内容。

1999 年，W3C 决定终止 HTML 的制定工作，转而制定 XHTML。一切都很完美，直到少数人注意到从 XHTML 升级到 XHML2 的升级工作几乎脱离实际。XML 的标准要求浏览器一旦碰到错误，就停止工作。另外因为 W3C 正在起草一种比老式、简陋的 HMTL 更出色的语言，它不赞成 (deprecate) 使用 img 和 a 标签这类元素。

Opera 和 Mozilla 开发人员不认同这种做法，并于 2004 年给 W3C 提交了一份报告，该报告称："我们认为网页应用 (Web Applications) 是一个极为重要的领域，但当前技术并未为这一领域提供充分的支持。在多方制定的规范出来之前，单一厂商的解决方案存在的潜在风险在不断增大。"(译注：暗指 Adobe 的 Flash 技术 ?)

这份报告提了 7 条设计原则

◆ 向后兼容，并有一个清晰的迁移路线 (migration path)

◆ 明晰 (Well-defined) 的错误处理机制，类似 CSS(比如，忽略未知内容，继续执行)，相比之下 XML 错误处理机制过于 "苛刻"。

◆ 编程错误不应直接暴露给终端用户。

◆ 实用性 : 所有最终进入网页应用技术规范的性特性都必须实际的应用案例支撑。但反之则不成立：即所有类似的应用案例并不必然会新特性加入到技术规范中。

◆ 脚本支持已经已得到公认 (但是当有更方便的标签可满足需求时，应避免使用脚本。)(译者：类似表单输入数据验证。)

◆ 避免针对特定设备的规范。

◆ 制定过程必须开放。(网络本身从开放式发展中受益颇多。邮件列表，存档，规范草稿应一直对公众开放。)

该报告遭 W3C 的拒绝，因此 Opera 和 Mozilla 以及后来的苹果继续维护着一个叫做网络超文本应用程序技术工作组 (Web Hypertext Application Technology Working Group，简称 WHATWG) 的邮件列表 (Mail list)。

制定他们用以验证概念 (proof-of-concept)

规范内容。这份规范对 HTML4 表单规范进行了扩充，在伊恩 · 希克森 (Ian Hickson) 的不断校订中，这份规范最终成为一份名叫网页应用程序 1.0(Web Applications 1.0) 的规范。后来伊恩 · 希克森离开 Opera，加入 Google。

在 2006 年，W3C 终于意识到自己的错误，决定重新启用 HTML，向 WHATWG 所要它的规范，并将其作为 HTML5 规范的基础

上面这些是史事资料。现在我们来看看一些流传甚广的流言。

流言“在 2012(或 2022) 年之前，俺是用不上 HTML5 的了。”

这一流言从 HTML5 进入到 W3C 流程候选推荐阶段 (Candidate Recommendation，简称 REC) 的项目日期所误传开来。官方 Wiki 上写道：

如今一个规范要成为候选推荐标准 (REC)，它需要具备百分之百的可实施性 (interoperable implementations)，只有成功通过上万项的测试案例 (Test Case) 后才能验证这点 (据保守估计，整个规范可能需要进行 2 万项测试)。当你在心里默算写这些测试案例需要多少时间，实施每个新特性又需要多少时间时，你就会明白 HTML5 规范制定的时间跨度为什么这么长了。

因此，按此说法，在你能在两大浏览器中用上所有的功能之前，HTML5 的规范是没有最终定稿的。

当然，真正重要的一小部分 HTML5 的特性已得到浏览器的支持，任何浏览器的支持情况汇总表单都会在一周之内过时，因为浏览器制作厂商的创新速度非常之快。另外，许多 HTML5 的新特性也通过 JavaScript 脚本在不支持 HTML5 的老浏览器中得以重现。Canvas 属性在所有新浏览器中得到支持，其中包括 IE9，另外在老的 IE 浏览器中，通过 excanvas 库，我们也可以模拟 Canvas 的特性。而音频和视频标签效果，我们则可以通过 Flash 在旧的浏览器中实现。

HTML5 在设计上就可以优雅降级，因此运用一些 JavaScript 代码和创意，HTML5 的所有功能都可以在老浏览器上实现。

“俺的浏览器支持 HTML5，你的不支持。”

这一流言认定 HTML5 是一个整体不可分割的标准。但实际上不是。正如前文所说，HTML5 是一组新特性的组合。因此，短期来讲，你不能说一个浏览器支持了 HTML5 的所有内容。而当浏览器能做到这点时，浏览器本身已经无关紧要了，因为那时我们将被新一代的 HTML 语言所震撼。

感觉 HTML5 乱的一塌糊涂，是吧？看看 CSS2.1，这么多年了它都是一个尚未最终完成的标准，但是我们每个人无时不在用它。我们用 CSS3 轻松添加圆角，这点很快就会得到所有浏览器的支持，虽然 CSS3 的其他特性尚未得到所有浏览器的支持。要提防那些浏览器“评分”网站。这些网站测试的内容经常与 HTML5 无关，比如 CSS，SVG，甚至是网页字体 (web fonts)。你手头需要完成的工作才是要紧的，你客户受众浏览器所支持的技术才是用得上的技术。

HTML5 实际上正式认可了一些常见的书写错误 (Tag Soup)

HTML5 在语法方面要比 XHTML 松散很多：比如，你可以用纯大写或小写字母书写标签，

有关 HTML 5 的流言与真相 III

甚至大小写混用也无妨。你无需对 img 这类的标签做自封闭处理 (self-close)，因此下面这两种写法都是合法的：

<img src=" nice.jpg" >

标签属性也无需用双引号括起来，因此下面这两种写法都是合法的：

<img src=" nice.jpg" >

<img src=nice.jpg>

使用大写或小写 (甚至混用) 字母都可以，所以下面三种写法也都是合法的 :

<IMG SRC=nice.jpg>

<img src=nice.jpg>

<iMg SrC=nice.jpg>

这与 HTML4 毫无差异，但是如果你用习惯了 XHTML，你碰到这种写法时还是会很震惊的。现实中，如果你使用 HTML 和文本内容书写页面，而非使用 XML(你极有可能是混用文本和 HTML 书写页面的，因为 IE8 并不能真正的渲染 XHTML 页面)，那么上述细微差别也无关紧要：浏览器会忽略尾部的斜杠，双引号，以及大小写。

HTML5 语法看似松散，但实际的解析规则要严格的多。因而 HTML5 中，常见的书写错误 (Tag Soul) 将不复存在 ;HTML5 的规范对这些无效标记做精确的描述和定义，因此所有遵循规范的浏览器都会生成同样的文档对象模型 (DOM)。如果你曾写过 JavaScript 来遍历 DOM，那么你就会对 DOM 不一致所带的恐怖经历有所体会。

但这种修正不应导致无效代码泛滥。HTML5 为你创建的 DOM 可能并不是你想要的那个，因此对书写的 HTML5 代码进行验证仍然至关重要。随着新特性的大量涌入，对细小语法错误的忽视会让你的脚本失效，或是 CSS 样式出错，这也是我们为什么需要 HTML5 验证器的原因之所在。

HTML5 远不仅仅只是让一些常见的书写错误合法化，而且让这些常见错误 (Tag soup) 成为历史。赞。

“我需要把我的网站从 XHTML 转换 HTML5。”

HTML5 对松散语法的包容性时候敲响的了 XHTML 的丧钟吗？制定 XHTML2 规范的工作组已经解散，对吧。

没错，XHTML2 的工作组在 2009 年年末的时候解了。他们起草的这个规范是用来与 HTML5 竞争的，但尚未得到执行实施，然而，同时保留两队人马是对 W3C 组织资源的一种浪费。另外 XHTML1 已经是一个业已完成的规范，得到所有浏览器的广泛支持，并在必须的时限内仍将得到所有浏览器的支持。因此你用 XHTML 书写的网站也将免受折腾之苦。

HTML5 将会干掉 XML

根本不会，如果你需要使用 XML 而是 HTML，你可以选用 XHTML5，它几乎包含所有 HTML5 的优点，只是要必须遵循严格 XHTML 语法 (比如，要标签属性中的双引号不能省，自封闭元素的末尾斜杠不可胜，必须用小写字母书写标签等等诸如此类。)

现实情况是 XHTML5 并不完全包含所有 HTML5 的特性。譬如 <noscript> 就失效了。但你想想，这古董玩意儿还有人在用吗？ ■未完内容请访问 http://developer.51cto.com/art/201106/269603.htm

.NET开发者应该关注HTML5五理由

如果您是微软客户端开发者致力于 C# 和 XAML, 您可能会为微软 Windows 8 投奔 HTML 5 与 JavaScript 而愤怒。这可以理解，但不得不承认这是一个新的世界。HTML 5 将成为下一代 Web 开发标准，下文就为你列举你应该关注 HTML 5 的五大理由。

微软近期在 D9 和 Computex 2011 大会上演示了 Windows 8，普通用户对于 Windows 8 的全新界面和触摸功能是欣喜不已，但对另一群人来说就是 2012，他们就是 Silverlight 和 .Net 开发人员，在 D9 大会上，微软演示了 Windows 8 如何为应用程序整合了一个以触摸操作为主的用户界面，这些应用都是基于 HTML 5 和 JavaScript 开发的。微软似乎将 Windows 8 的关注点都放在这个新的 Web 标准开发平台 HTML 5，并用它来驱动全新的基于卡片 (Tile) 的触摸界面，却全然忽略了现有的 Windows 开发平台：Silverlight 和 .Net。这让 Silverlight 和 .Net 开发人员不禁感慨伤不起！

如果您是微软客户端开发者致力于 C# 和 XAML, 您可能会为微软 Windows 8 投奔 HTML 5 与 JavaScript 而愤怒。这可以理解，但不得不承认这是一个新的世界。HTML 5 将成为下一代 Web 开发标准，下文就为你列举你应该关注 HTML 5 的五大理由。

1. HTML 5 代表 Web 开发者的一次转型

HTML 5 作为下一代的 Web 开发标准，其特性已经慢慢地出现在主流的浏览器中，这种新的 HTML 将会让浏览器不必再依赖 Flash、QuickTime、Silverlight 等插件，也简化了原来需要大量 JavaScript 才能达到的效果。然而作为 Web 开发人员正面临着一次转型，首先心里上的转型，接受 HTML 5; 技术上的转型，学习掌握 HTML 5，知道如何把 HTML 5 转化为各种 Web 应用，能够做到将现有的 Web 应用过渡到 HTML 5。有挑战也有机遇，HTML 5 相对 Flash 来更简单易学，Web 开发者又重新站在同一起跑线上，共享胜果。

2. HTML 5 在移动 Web 应用的优势

移动领域开发需要面对多平台多终端的现实，HTML5 是相对成熟的可以适应复杂界面的解决方案，可以有效地降低开发成本和周期。Android 和 iOS 手机的兴起，也加速了 HTML5 在移动设备的普及。移动浏览器的不断升级，给 HTML 5 在移动 Web 方向的发展提供源源不断的动力。也随着设备性能的不断提高，移动 Web 应用的能力也渐渐逼近客户端应用。

另外未来几年，移动应用开发中的 HTML5 技术的调试工具无疑会变得更加重要，它可以解决大部分开发人员 80% 的工作量。你想要用 Objective-C 改变你的界面设计吗？编辑，再编译，运行。重复这三个步骤直到你满意为止。如果再编译步骤很多，这可能会耗上一天的时间。用 HTML5 技术去实现？

五理由 .NET 开发者应该关注 HTML 5 II

用 weinre 编辑一些 CSS 属性并测试，你甚至不用关闭应用，你就可以继续调试。一定程度上，你还可以在桌面浏览器调试你的 HTML5 手机应用。但相信我，你的应用产品最终可能只会在移动设备上爆发一大堆漏洞而已，所以你必须得使用 weinre。

3. 一个 Windows 应用程序市场潜力巨大

苹果公司刚刚宣布靠出售应用程序已经赚了 14 亿美元，而且会投资 25 亿美元用来开发应用程序。现在有 200 万的 iOS 设备来运行这些应用程序。如果你认为移动市场是巨大的，那么桌面呢？相比之下，自 2009 年 7 月 22 日微软 Windows 7 发布以来，微软目前已经售出了 4 亿份 Windows 7 拷贝，目前每秒钟就有 7 份 Windows 7 副本被售出。当微软发布了一套新的基于 Windows 8 的触控中心界面。依照演讲稿及相关的新闻稿所示，构建该界面仅需在 Windows 8 上通过 HTML5 和 JavaScript 的技术即可实现，同时，HTML5 和 JavaScript 可以使用 PC 的全部功能。这时候我们理应欣喜若狂，一个巨大的蛋糕正摆在了我们面前，而且是移动市场的数倍。

4. HTML 5 游戏的重大变革

appMobi 宣布了一项新的技术：DirectCanvas，号称可以为手机 HTML5 游戏缓慢的图形渲染提升 500% 的速度。DirectCanvas 的设计是为了为 HTML5 元素加速，让游戏的图形和动画更流畅，换句话说，它让你的游戏速度更快出问题更少，点击查看这项技术。

5. 开发者应该未雨绸缪

这说起来容易，但 HTML 5 是不是真的准备好了呢？它的一切都是新的，每天都在变化，最终的标准也只会在未来的几年内发布，但未来不会只见树木不见森林。

HTML 5 不是一场雷雨，而是一个浪潮，聪明的开发商不会在被大水淹没才会去学习游泳，而作为开发者更应该未雨绸缪■

为 HTML 5 的未来制定学习计划

以下就你希望学习来步入此门槛的技术：

HTML 5：尽管 HTML5 作为标准可能尚未完全确定，但是到了这个阶段变动不会太大了。目前，它已经是可用的，并在相当一些浏览器上有了实现。你现在可以开始学习它了。

CSS：如果你对 CSS 还不熟悉，现在正是学习它的大好时机。浏览器支持还在不断完善，且现在 IE6 在市场的份额已经很少，对此开发人员已经觉得可以放心地忽略不计。

Web services： 每一种主流的服务器端开发语言都拥有一个框架或一套库，以便容易地生成 web 服务，如 .NET 中的 Windows Communication Foundation (WCF)。理解这些基础应该不需要花费太多的学习精力。你也许特别希望学会如何生成 JSON 的输出，此物正在迅速变成 Web 应用的通用语。还有，确保你理解了 RESTful Web 服务。在现代的开发环境下，相对于 SOAP 来说，尽管它们可能需要花费更多的功夫，但是其可访问性可用性也要高得多。

JavaScript： 相对于传统的 ASP.NET 或类似开发需求来说，新的应用开发范式需要多一点 JavaScript 方面的知识。

jQuery： jQuery 已经成为可用的客户端开发框架；它似乎可用包办一切。在利用一组扩展的插件的情况下，如果你想玩点 UI 的花样，jQuery 能帮你忙。

从零开始构建HTML 5 Web页面

HTML 5 是时下 Web 开发领域炒得火热的一个术语，是的，很多人都看好它，也有很多业内知名公司开始正式使用 HTML 5 重新构建自己的网站，如 YouTube 开始使用 HTML 5 视频，Google 已经弃用自家的 Gears，开始全面拥抱 HTML 5 实现离线解决方案，各大浏览器厂家也纷纷开始支持 HTML 5，连被人诟病的微软也声称要在 IE 9 中增加对 HTML 5 的支持。本文打算为大家详细介绍一下如何构建一个完整的 HTML 5 Web 页面，以加深对 HTML 5 的理解。

HTML 5 有何不同?

首先我们要明白 HTML 5 是新的语义结构标记，包括画布，离线存储规范和一些新的内联语义标记，但由于客观原因（主要是浏览器支持的原因），我们不得不限制标记的讨论范围，如画布，离线存储，原生视频或地理定位 API 等，还不是所有的浏览器都支持。

由于新的 HTML 5 标记大都是结构性的，它们的行为与块元素有些类似，为了帮助大家加深对 HTML 5 的理解，我将在下面的内容使用一些新的结构元素。

每个人都应该记住的 doctype（文档类型）

要创建 HTML 5 Web 页面的第一件事情是使用新的 doctype，你一定记住了 HTML 4 或 XHTML 1.x 的 doctype，当我们要从旧的文档通过复制粘贴到新的文档中，必须要修改 doctype，请记住，下面就是 HTML 5 的 doctype：

```
<!DOCTYPE html>
```

还是很容易记住吧，而且也不区分大小写，与现在广泛使用的版本要简单得多了，而且保持了向后兼容。

语义结构

在深入标记前，我们先初略看一下一个 Web 页面的大致结构吧。

```
<html> <head> ...stuff...  </head>
  <body>
      <div id="header">
         <h1>My Site</h1> </div>
      <div id="nav"> <ul>
            <li>Home</li>
            <li>About</li>
            <li>Contact</li>  </ul> </div>
      <div id=content>
<h1>My Article</h1> <p>...</p> </div>
      <div id="footer"> <p>...</p>
      </div> </body></html>
```

在上一页的例子中，我为所有 DIV 标记增加了 ID，相信大多数 Web 设计师都很熟悉这种做法了，这么做有两个目的，首先，ID 提供了一个钩子，通过它可以对页面的特定部分应用样式，其次，ID 作为一种原始的，伪语义结构，智能解析器将查找标签上的 ID 属性，并尝试猜测其含义，但这是一件很困难的事情，因为每个网站的 ID 可能都不一样。

于是就有了增加新标签的想法，HTML 5 的创造者们就设计了一些新的元素，下面我们就来看看 HTML 5 中新增的一些关键的结构性标记。

<header>

这个标记计划用来描述一节或一个完整 Web 页面的介绍性信息，<header> 标记可以包括所有的通常放在页面头部的标志，若你在页面中使用了 ，它将被 <header> 取代。

<nav>

这个元素的含义就不说了，你的导航元素就放在这里，如主站点导航，但在某些情况下也可能有页面导航元素，HTML 5 的创建人 WHATWG 最近修改了 <nav> 的解释，展示了如何在一个页面上使用两次。有关 nav 更多的讨论，可以参考：http://www.zeldman.com/2009/07/13/html-5-nav-ambiguity-resolved/。

简单说来，如果你在页面中使用了 标记来容纳导航元素，那么你可以使用 <nav> 进行替换。

<section>

这个可能是最模糊不清的标记了，按照 HTML 5 规范的解释，一个 section 是一个有主题的内容组，前面通常有一个 header 标记，后面通常跟一个 footer 标记，如果需要，section 也可以嵌套使用。

在我们上面的例子中，标记为“content”的 DIV 是 section 的一个很好候选者，在这个 section 中，根据内容的不同，我们可能有更多的 section。

<article>

根据 WHATWG 的注释，article 元素是将 section 进行打包形成一个文档或网站独立的部分，例如一篇杂志或报纸文章，或一篇博客文章。

记住，在一个页面中可以有多个 article 元素，例如一个博客首页可能会有 10 多个 article 元素，article 也可以进入 section 元素，因此你在嵌套使用时需要小心，可能一不小心就会出错。

<aside>

另一个模糊不清的标记是 aside，这个元素表现的是与文档主要文本流无关的内容，也就是相当于一个括号备注，脚注，引用，注释，或者说是类似于侧边栏的东西，根据 WHATWG 的注释，<aside> 可以用于所有这些情况。

<footer>

Footer 的含义也非常清楚，它可以用在 section 中，也可以用于一个页面的底部。

全部放在一起

新标记的样式

与旧浏览器的兼容（该部分有省略，详细版本参见 http://developer.51cto.com/art/201002/184382.htm）

现在已经可以使用 HTML 5，但应该用吗？

答案很简单：是的！

但这也要根据网站的性质做出调整，例如，假设你要重构 CNN 主页，那可能不太现实，最好还是等浏览器的支持更好一些再说，但如果你是在翻新你的博客系统，那么你可以一试，如果你使用的 WordPress，目前已经出现了一些插件可以帮助你，这里就有一个 HTML 5 的 WordPress 主题。

你也可以去 HTML 5 画廊（http://html5gallery.com/）瞧瞧，因为它全部是用 HTML 5 构建的，可以看看其源码，加深对 HTML 5 标记的理解。还可以继续关注 51CTO.com 的 HTML 5 专题，我们会持续更新关于 HTML 5 的技术应用和资讯报道。

如果你还有点犹豫不决，那你去看看 Google 的主页，已经是 HTML 5 了，保险一点的话，你可以使用 JavaScript 声明这些新标记进行使用。HTML 5 的标记远不止这些，希望本文能消除你的疑虑，大胆使用 HTML 5，只有使用的人多了，这个规范才能真正见效。■

项目经理该如何培养优秀的程序员

虽然现在已经不是项目经理了，跳来新公司已经变成程序员级别了，但是不免拿现在的经理和以前的我在职责和处事作风做些比较。

果然，位置不同，看事情的角度便不同。发现以前自己有很多的地方处理不是很好，也能够发现现在的经理一些值得商榷的地方。

本来今天只想谈谈一些经验和让大家避免做错事和如何做一个好的程序员，至少知道项目经理关心什么，但是刚刚回忆了下，前后跟过真的很多的项目经理，从日系外包到国网项目，再到现在这个在财务的公司，有过不少项目经理或者说项目负责人，一时之间，比较清晰的思路，又变得不清晰起来。

项目管理，名义上是项目的管理，但是其实是两方面，一个是对项目，一个是对人。对项目是硬件，对人是软件。

硬件呢，比较死，当然可以用活。软件，其实是很重要的，正常的职业经理，其实是管人，而人是做事，和人共事是一件相当之麻烦的事情，远比敲代码麻烦得多。

项目经理对于自己的组员，应该如何态度对待，这个问题，看书上，那多得是天花乱坠，几乎我看得头昏眼花，其实归根究底，就是沟通。

沟通，只要是本管理的书就说沟通，沟通来沟通去，我就没明白啥是沟通。说个词，谁不会，怎么沟通？如何沟通？沟通后要怎么让它产生作用？终究我们不是演讲大师，做程序员的有几个口才好的，口才好就去做销售啦；

一听讨论业务，嗓门最大的那个，一定就是这个项目的项目经理。一般程序员升上去的，一讨论业务，嗓门直线上升。我以前也这样，当时我没发现，但是有人和我说过，没在意。到了新公司看程序员升到项目经理后，嗓门大开。原来当项目经理可以开嗓，那些歌星该来体验下。

好像跑题啦，我们继续沟通的话题。既然口才不到位，那是不是不能沟通啦？废话，当然能！

怎么勾呢！关怀呗。怎么关怀呢！用心呗。怎么用心呢！巴结人你不懂啊！这里说，其实关心你的同事，其实很重要，作为一个领导开发的人，不能只关注进度，及质量，人才是最重要的；

举个例子，你有同事生病了，当然他可能找理由干其他事。你要关心他，怎么关心？问句话？发个短信？是打电话，如果他回家休息很多天，那你要有礼物；明白吗？

如果你的同事觉得不舒服，你要问问他。然后呢，你安慰下，就没事啦，让他请假？他不走呢？你怎么办，放任不管还是放屁，是买药去！

反正软件公司对中间出门买东西，管得不严，你出去买个药，算多大事！你的药几块钱，换来的可不是几块钱回报，哥们，这生意不亏！既然关怀了，我们当然要主动关心，你的同事。

我拿新员工举例，主要我现在处于这位置。

大家工作过几年，换过几次工作，会有一些体验，在一个公司最开始的那几天是最难熬的，这是为什么呢？因为不熟吗？

新员工上班 2 个星期内辞职率是最高的，这是我在以前公司人力资源演讲报告里面挺的，我当时对人力资源比较感兴趣，就要了份，学习下。

项目经理该如何培养优秀的程序员 II

现在的软件公司，有几个普遍的不好现象，笔者最讨厌的，但是也是最无奈的。

1. 新员工上班，没人管，没人理

你的组长或经理根本不想理你，总说是忙，自己弄去。结果部署个项目，还要自己研究少包没，一个代码部署运行，搞了一个星期，还被他的组长训。顺便说下，这是真事，那个不好的组长是笔者兄弟，笔者当时是另外一个组的组长，我俩关系很好。这个人什么都好，就是没耐心教别人，话总说一两句，让你自己研究吧。在加上他真的很忙，后来说真的，是真看不下去了，我去找我那兄弟说了这事，有空也去帮帮那人。叹口气，也就是那是个新人，刚毕业，没什么经验，他是忍了，但是这要是一般人估计早撂挑子了。

对待新人，不要一份漫不经心的态度，而是要加倍的关心，关怀，主动去问遇到什么问题，程序员没几个是主动能力比较好的，遇问题特别刚到公司，都不想被瞧不起，总是硬抗，这太坑人了，万一他没抗住，那他就真走了，话说不一定是能力问题，你都没和人讲清楚，人家怎么弄。

如果下属同事，有困难不能解决，又忍了很多天，最后爆发出来，这问题不是他的问题，一定是你的问题。

2. 新员工上班不受重视

深有体会，当年我做得一样不好，不过现在我知道如何补救啦。新员工上班，最大的一个困难就是适应，适应环境。环境是什么？不是桌子，椅子，大可乐，我喜欢喝可乐。咳，又跑题了，回来！

环境主要在于气氛，有的公司气氛很压抑，有的很活跃，这个主要看公司的人，我个人喜欢活跃的气氛，比较讨厌死气沉沉的，做事要开心，开心才做事。这个时候，新员工的心理状态是最不稳定的，他的精神也紧绷的，这个时候你要做的如何让他紧绷的神经，松弛下来。

我们回到项目经理冷漠的问题。

项目有大有小，大的几十开发人员，小的几个开发人员，因此管理手段是不一样的，大的项目用管理过程，走流程，走规章制度，小的就要走人情，这是必然的。

反正无论哪个，都离不开人。在项目经理周围一定存在小集体，这个小集体一般在项目经理周边，故很多时候项目经理只关注身边人(典型)。而忽略其他的成员，简单说就是厚此薄彼，正常员工都这样，你一新来的，你算个什么啊！给我哪凉快哪呆着去。于是乎，很多时候，项目经理不知道成员在做什么，成员不知道项目经理下一步要做什么，导致成员懒散的个性，这是必然。

这里还有一个问题，如果有组长，项目经理是否应该还要接触相关的组员，是否接触组长就够了？这个问题其实，一直困扰很久一段时间，也没有太好的办法，这里抛出了，大家有啥好想法，就说说。这个问题导致一些问题，一些组长是到三十而立的年纪，用年头混上来的，当时我也是组长，但是我可不是混上来的，虽然很年轻。

这些年长的组长，对于工作其实不是特别上心，对于工作之外的事情，倒是很积极，人也圆滑的可以，这些组长对于下属，绝对是爱答不理，告诉你事情呢，也是一知半解。而这时候，项目经理，如果只对组长负责，你如何知道项目真正的进展情况，质量问题都是组长负责的。项目经理和一个在紫禁城里发号施令的皇帝有区别吗？

他完全不了解项目真正的情况，如果他去

项目经理该如何培养优秀的程序员 III

接触组员，又是跨级，避免不了跨级小报告的问题，这个又是管理上的问题，这个问题，苦思 N 久，没想到问题，那时候，我还是组长，所以这问题，也不需要真的面对，故此也没有特别深究，一直扔在那了。

继续我们的冷漠问题，项目经理冷漠可能很多原因，事情忙，人太多，进度赶，各种问题存在，由此估计不到每个人。

首先，如果你是项目经理，你扪心自问，你了解你每一个组员吗？你知道他们在想什么吗？你知道他们需要什么吗？你不跟他们接触，他能知道什么？

记得以前学项目管理的时候，有一个方式被现在的公司都快用烂啦，就是让你的组员写一份职业规划及发展方向说明书，这个事，我也干过。

为啥说这个事呢？问一句，哪个项目经理看完过所有组员职业规划？就算你看完了，你有考虑过吗？我就算你有考虑过，你看那么多份报告，你记得住吗？这东西，形式主义居多，反正我就看了几份，就记不住谁是谁的。

我们这么来说，你要让你的组员来写份文档来告诉你，他的目标，你不觉得这是你的失败吗？

上面都说沟通啦，你竟然没看出来你组员的想法，这玩意说明你根本没将你组员放在心上。

你如果想用文档来让组员告诉你他的想法，那么就证明，你不是一个合格组长或项目经理。

至于怎么交流，我说一个我诀窍好了，和组员们做好一百件好事，远远不如带着你的组员做一件坏事，来的感情稳固。其实解决项目经理冷漠的方法，其实有很多种。

对于项目经理不知道如何与组员沟通，我在管理学的书上看到一招，挺好，但是没用过。因为看到时，我已经变回程序员了。

是这样，他说人呢，分为如下很多种：

1. 现实型员工 – 动手能力强，思考能力差；– 程序员；擅长工具。

2. 研究型员工 – 动手能力差，思考能力强，抽象能力强 – 系统分析，架构师；擅长事，物。

3. 传统型员工 – 会压抑自身的个性，以职位及工作决定性格 – 会计，出纳，金融顾问；擅长事。

4. 进取型员工 – 个人冲劲特别足，有时间就会充实自己 – 销售，经理。

5. 社交型员工 – 喜欢和人打交道 – 经理。

6. 艺术型员工 – 喜欢出头，喜欢成为焦点。

人的性格决定很多的事情，故此我们按照他的类型，来确定他是怎么样的人；

然后，我们设计一个时间点，比如说：

我组员，张三 – 现实型，李四 – 研究型；

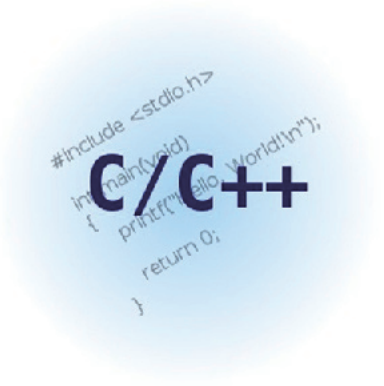
通过类型，我们设计时间点，张三 –2 天，李四 –1.5 天。什么意思呢，就是我们潜意识规定自己，在 2 天内，至少要主动跟张三交流 1 次，在 1.5 天内，至少要主动跟李四交流 1 次；

交流的方式，你就自己定吧，嘘寒问暖也好，工作问题也罢，你怎么找小姑娘搭讪的，就怎么跟他们交流。■ （作者：刃月）

■ 特别推荐：51CTO 电子杂志《Linux 运维趋势》第 10 期发布，订阅地址请访问下面网页：

http://os.51cto.com/art/201107/274672.htm

掀起C++ 11神秘面纱



C++ 之父说 C++11 就像一个新语言，的确，C++11 核心已经发生了巨大的变化，它现在支持 Lambda 表达式，对象类型自动推断，统一的初始化语法，以及最重要的右值引用。

在这篇文章中，我将介绍 C++11 标准中发生的最大变化，以及为什么应该引起注意，正如你将看到的，线程库不是唯一的变化，新标准采纳了数十位专家的意见，使 C++ 变得更有意义。正如 Rogers Cadenhead 指出的那样，它们就像迪斯科、宠物石和长胸毛的奥运游泳选手一样不可思议。

首先，让我们看看 C++11 核心语言的一些突出特性。

Lambda 表达式

Lambda 表达式允许你在本地定义函数，即在调用的地方定义，从而消除函数对象产生的许多安全风险，Lambda 表达式的格式如下

```
[capture](parameters)->return-type {body}
```

[] 里是函数调用的参数列表，表示 Lambda 表达式的开始，让我们来看一个 Lambda 例子：

假设你想计算某个字符串包含多少个大写字母，使用 for_each() 遍历一个 char 数组，下面的 Lambda 表达式确定每个字母是否是大写字母，每当它发现一个大写字母，Lambda 表达式给 Uppercase 加 1，Uppercase 是定义在 Lambda 表达式外的一个变量：

```
int main()
{  char s[]="Hello World!";
int Uppercase = 0; //modified by the lambda
for_each(s, s+sizeof(s), [&Uppercase] (char c) {  if (isupper(c))
Uppercase++;  });
cout<< Uppercase<<" uppercase letters in: "<< s<<endl;  }
```

这是因为，如果你定义的函数主体被放置在另一个函数调用内部，[&Uppercase] 中的“&”记号意味着 Lambda 主体获得一个 Uppercase 的引用，以便它能修改，如果没有这个特殊记号，Uppercase 将通过值传递，C++11 Lambda 表达式也包括成员函数构造器。

自动类型推断和 decltype

在 C++03 中，在声明对象时，你必须指定对象的类型，然而，在许多情况下，对象的声明包括在初始化代码中，C++11 利用了这个优势，允许你声明对象时不指定类型：

```
auto x=0; //x has type int because 0 is int
auto c='a'; //char
auto d=0.5; //double
auto national_debt=14400000000000LL;//long long
```

相反，你可以声明下面这样的迭代器：

```
void fucn(const vector<int> &vi)
{vector<int>::const_iterator ci=vi.begin();  }
```

掀起 C++ 11 的神秘面纱 II

关键字 auto 不是什么新生事物，我们早已认识，它实际上可以追溯到前 ANSI C 时代，但是，C++11 改变了它的含义，auto 不再指定自动存储类型对象，相反，它声明的对象类型是根据初始化代码推断而来的，C++11 删除了 auto 关键字的旧有含义以避免混淆，C++11 提供了一个类似的机制捕捉对象或表达式的类型，新的操作符 decltype 需要一个表达式，并返回它的类型。

```
const vector<int> vi;
typedef decltype (vi.begin()) CIT;
CIT another_const_iterator;
```

统一初始化语法

C++ 至少有 4 个不同的初始化符号，有些存在重叠，括号初始化语法如下：

```
std::string s("hello");
int m=int(); //default initialization
```

在某些情况下，你也可以使用“=”符号进行初始化：

```
std::string s="hello";  int x=5;
```

对于 POD 聚合，你还可以使用大括号：

```
int arr[4]={0,1,2,3};
struct tm today={0};
```

最后，构造函数使用成员进行初始化：

```
struct S { int x; S(): x(0) {} };
```

显然，这么多种初始化方法会引起混乱，对新手来说就更痛苦了，更糟糕的是，在 C++03 中，你不能初始化 POD 数组成员，POD 数组使用 new[] 分配，C++11 使用统一的大括号符号清理了这一混乱局面。

```
class C
{  int a;  int b;
   public:  C(int i, int j);  };
C c {0,0}; //C++11 only. Equivalent to: C c(0,0);
int* a = new int[3] { 1, 2, 0 }; /C++11 only
class X {   int a[4];  public:  X() : a{1,2,3,4} {} //C++11, member array initializer  };
```

关于容器，你可以和一长串的 push_back() 调用说再见了，在 C++11 中，你可以直观地初始化容器：

```
// C++11 container initializer
vector vs<string>={ "first", "second", "third"};
map singers = { {"Lady Gaga", "+1 (212) 555-7890"},
{"Beyonce Knowles", "+1 (212) 555-0987"}};
```

C++11 支持类似的数据成员类内初始化：

```
class C
{  int a=7; //C++11 only
   public:   C(); };
```

Deleted 和 Defaulted 函数

一个表单中的函数：

```
struct A
{  A()=default; //C++11
   virtual ~A()=default; //C++11   };
```

被称为一个 defaulted 函数，“=default;”告诉编译器为函数生成默认的实现。Defaulted 函数有两个好处：比手工实现更高效，让程序员摆脱了手工定义这些函数的苦差事。

与 defaulted 函数相反的是 deleted 函数：

```
int func()=delete;
```

Deleted 函数对防止对象复制很有用，回想一下 C++ 自动为类声明一个副本构造函数

掀起 C++ 11 的神秘面纱 III

和一个赋值操作符，要禁用复制，声明这两个特殊的成员函数 =delete 即可：

```
struct NoCopy { NoCopy & operator =( const NoCopy & ) = delete;
NoCopy ( const NoCopy & ) = delete; };
NoCopy a; NoCopy b(a); //compilation error, copy ctor is deleted
```

nullptr

C++ 终于有一个关键字指定一个空指针常量了，nullptr 取代了有错误倾向的 null 和文字 0，这两个被用来作为空指针替代品已经有很多年的历史了，nullptr 是一个强类型：

```
void f(int); //#1
void f(char *);//#2
//C++03
f(0); //which f is called?
//C++11
f(nullptr) //unambiguous, calls #2
```

nullptr 适用于所有指针类别，包括函数指针和成员指针：

```
const char *pc=str.c_str(); //data pointers
if (pc!=nullptr)
  cout<<pc<<endl;
  int (A::*pmf)()=nullptr; //pointer to member function
void (*pmf)()=nullptr; //pointer to function
```

委托构造函数

在 C++11 中，构造函数可以调用相同类中的其它构造函数：

```
class M //C++11 delegating constructors
{ int x, y; char *p;
public: M(int v) : x(v), y(0), p(new char [MAX]) {} //#1 target
M(): M(0) {cout<<"delegating ctor"<
```

构造函数 #2，委托构造函数，调用目标构造函数 #1。

右值引用

C++03 中的引用类型只能绑定左值，C++11 引入了一种新型引用类型，叫做右值引用，右值引用可以绑定左值，例如，临时对象和字面量。增加右值引用的主要原因是 move(移动) 语义，它和传统的复制不一样，移动意味着目标对象偷窃了源对象的资源，留下一个状态为“空”的资源，在某些情况下，复制一个对象代价既高又没有必要，可以用一个移动操作代替，如果你想评估移动带来的性能收益，可以考虑字符串交换，一个幼稚的实现如下：

```
void naiveswap(string &a, string & b)
{ string temp = a; a=b; b=temp; }
```

像这样写代价是很高的，复制字符串必须分配原始内存，将字符从源位置复制到目标位置，相反，移动字符串仅仅是交换两个数据成员，不用分配内存，复制 char 数组和删除内存：

```
void moveswapstr(string& empty, string & filled) { //pseudo code, but you get the idea
size_t sz=empty.size();
const char *p= empty.data();
//move filled's resources to empty
empty.setsize(filled.size());
empty.setdata(filled.data());
//filled becomes empty
filled.setsize(sz); filled.setdata(p); }
```

掀起 C++ 11 的神秘面纱 IV

如果你实现的类支持移动，你可像下面这样声明一个移动构造函数和一个移动赋值操作符：

```
class Movable
{ Movable (Movable&&); //move constructor
  Movable&& operator=(Movable&&); //
move assignment operator  };
```

C++11 标准库广泛使用了移动语义，许多算法和容器现在都为移动做了优化。

C++11 标准库

C++ 于 2003 年以库技术报告 1(TR1) 的形式经历了重大改版，TR1 包括新的容器类 (unordered_set，unordered_map，unordered_multiset 和 unordered_multimap) 和多个支撑正则表达式、元组和函数对象封装器等的新库。随着 C++11 标准获得通过，TR1 和自它首次发布以来新增的库被正式纳入标准的 C++ 标准，下面是 C++11 标准库的一些特性：

线程库

站在程序员的角度来看，C++11 最重要的新功能毫无疑问是并行操作，C++11 拥有一个代表执行线程的线程类，在并行环境中用于同步，async() 函数模板启动并行任务，为线程独特的数据声明 thread_local 存储类型。如果你想找 C++11 线程库的快速教程，请阅读 Anthony William 的 “C++0x 中更简单的多线程”。

新的智能指针类

C++98 只定义了一个智能指针类 auto_ptr，它现在已经被废弃了，C++11 引入了新的智能指针类 shared_ptr 和最近添加的 unique_ptr，两者都兼容其它标准库组件，因此你可以在标准容器内安全保存这些智能指针，并用标准算法操作它们。

新的算法

C++11 标准库定义了新的算法模仿 all_of()，any_of() 和 none_of() 操作，下面列出适用于 ispositive() 到 (first, first+n) 范围，且使用 all_of(), any_of() and none_of() 检查范围的属性的谓词：

```
#include <algorithm>  //C++11 code
//are all of the elements positive?
all_of(first, first+n, ispositive()); //false
//is there at least one positive element?
any_of(first, first+n, ispositive());//true
// are none of the elements positive?
none_of(first, first+n, ispositive()); //false
```

一种新型 copy_n 算法也可用了，使用 copy_n() 函数，复制一个包含 5 个元素的数组到另一个数组的代码如下：

```
#include
int source[5]={0,12,34,50,80};  int target[5];
//copy 5 elements from source to target
copy_n(source,5,target);
```

算法 iota() 创建了一个值顺序递增的范围，好像分配一个初始值给 *first，然后使用前缀 ++ 使值递增，在下面的代码中，iota() 分配连续值 {10,11,12,13,14} 给数组 arr，并将 { ‘a’,’b’,’c’} 分配给 char 数组 c。

```
include <numeric>
int a[5]={0};  char c[3]={0};
iota(a, a+5, 10); //changes a to
{10,11,12,13,14}  iota(c, c+3, 'a'); //{'a','b','c'}
```

C++11 仍然缺乏一些有用的库，如 XML API，套接字，GUI，反射以及前面提到一个合适的自动垃圾回收器，但 C++11 也带来许多新特性■

C++老矣,尚能饭否?

作者 / 彭凡

中国人读“C++”时存在两种读音,一种读作“C 加加”,另一种读作“C plus plus”。两种读音似乎都没错,只是遵从不同的标准罢了。如同 C++ 有两种读法一样,C++ 阵营甚至没有一个完整的标准,只有最符合和接近 C++ 标准的编译器 GNU GCC 4.6.1 和微软的 Visual Studio 2010 SP1。C++ 这个在 Tiobe 编程语言排行榜中跻身三甲的语言,为何连完整标准都没有?

C++ 之父 Bjarne Stroustrup 简历上只有一句话“C++ 缔造者”,这样的成就是荣耀无比的。但似乎 C++ 似乎过于低调,让很多 C++ 程序员都没有新消息可以接收。Java 有 Oracle 等大公司推动、.NET 有微软推动、HTML 5 有 Google 等推动、包括 PHP 有众多用户和社区推动,那 C++ 有谁在推动呢?

C++ 从 20 世纪 80 年代面世,经历了以下三个阶段的发展:

第一阶段从 1980 年代到 1995 年。这一阶段 C++ 语言基本上是传统类型上的面向对象语言,并且凭借着接近 C 语言的效率,在工业界使用的开发语言中占据了相当大份额;

第二阶段从 1995 年到 2000 年,由于标准模板库 (STL) 和后来的 Boost 等程序库的出现,泛型程序设计在 C++ 中占据了越来越多的比重性。当然,同时由于 Java、C# 等语言的出现和硬件价格的大规模下降,C++ 受到了一定冲击。

第三阶段从 2000 年至今,由于以 Loki、MPL 等程序库为代表的产生式编程和模板元编程的出现,C++ 出现发展历史上又一个新的高峰,这些新技术的出现以及和原有技术的融合,使 C++ 已经成为当今主流程序设计语言中最复杂的一员。

不管我们用的是 Borland 公司还是微软的 VC 环境,很少有程序员关心 C++ 的标准和版本问题。我们经常能听到 Java 或 .NET 程序员谈论 Java 7 或者 .NET 3.5 等版本的新特性,但有谁关心过 C++ 的新版本?

51CTO 记者随机对一些 C++ 程序员进行了调查,有 50% 的程序员表示对开发语言的新版本有兴趣,但不会用到实际工作中。只有当所在公司要求他们统一进化到新版本时,才会使用新版本。有 10% 的程序员会从繁忙的工作中抽出时间来学习新版本,并独自运用到开发工作中。剩下 40% 的程序员表示天天加班,根本没有时间看所谓的新版本,到时候现学吧。

这样的调查结果让我们知道了国内程序员对编程语言新版本的态度。从《掀起 C++ 11 的神秘面纱》中,我们听到了不少 C++ 的程序员对于 C++ 11 的悲观论调,甚至预言未来的 C++ 11 进不了编程语言排行榜的前十。

C++ 真的老了?

现在最好找工作的程序员是哪一类?根据 51CTO 编辑在智联招聘网站,以北京地区为范围,搜索到不同开发语言工作岗位数量如下:

PHP 5212、Java 11824、C++ 9252、ASP.NET 2383

可以看出 C++ 程序员的需求是旺盛的,甚至超越了因电子商务而红火的 PHP 程序员。C++

C++ 老矣，尚能饭否？ II

并不因为版本的问题而遭到抛弃，廉颇老矣，尚能饭否的应该是 VB 这样的语言。因为在智联招聘里，VB 这个关键词只有可怜巴巴的 827 个职位。

C++ 还是很多程序员的饭碗，大家一致用行动证明 C++ 并没有老去，并没有被人所遗忘。只是我们身在 C++ 中，并不太关心它的进步和发展，只要用 C++ 能处理好手头上的工作，哪管 C++ 11 的未来？

为什么新手不选择 C++？

很多新人喜欢 Java 或者 .NET，因为他们开发起来相对简单，甚至说高效。以往 C++ 优势在于对硬件资源的合理分配，特别是代码行超过三万以后，C++ 系统消耗优势更加明显，这也是为什么很多大公司热衷于 C++ 做大型开发。

但随着内存和 CPU 等硬件设备的降价，硬件资源和性能问题似乎不再成为程序员们考虑的问题。就好像 386、486 时代的 PC 用户还要为那几十 K 的内存优化费劲一样，C++ 程序员那穷尽极致的节约系统资源，已经被看做是费力不讨好，新手们自然不选择 C++ 了。

其实百度和《魔兽世界》都是 C++ 写出来的。

看 C++ 11 的新特性

C++ 没有垃圾回收机制，未来似乎也没有。如果你不小心没管住内存，那么就要对不起了，内存崩溃的情况不是不可能。在 Java 和 .NET 都支持垃圾回收之后，C++ 11 还是显得有些另类。当然我们也是猜测，这样对内存的精确控制，是不是出于对程序开发更加精确的考虑？

新的 C++ 11 里将会支持多线程开发，这个与目前多核 CPU 技术的发展密切相关，能极大的提高 C++ 开发成品的效率。这个新特性应该是与时俱进的改变，还是没有多少创新性。Visual Studio 2010 也已经实现 C++ 并行构建的功能。

Lambda 表达式也是此次 C++ 11 标准中最为人诟病的特性，甚至有 C++ 程序员直言“这 Lambda 太丑陋了，还不如 Java，尽管 Java 的 Lambda 也是极其丑陋。”也有清醒的 C++ 程序员认为 Lambda 表达式在大型项目中的应用还是不错的，新的语言机制会带来新的效率，尽管这会有些阵痛。关于 Java 7 与 C# 中 Lambda 表达式的优劣，请点击这里。

借用一位 C++ 程序员的例子来说明 Lambda 的正面作用，在 C++ 中，STL 的很多算法都要求使用者提供一个函数对象。例如 for_each 函数，会要求用户提供一个表明“行为”的函数对象。以 vector<bool> 为例，如果想使用 for_each 对其中的各元素全部赋值为 true，一般需要这么一个函数对象。

```
class filler { public:
void operator()(bool &i) const {i = true;} };
```

这样实现不但麻烦，而且不直观。而如果使用 lambda，则允许用户使用一种直观和见解的方式来处理这个问题。以 boost.lambda 为例，刚才的问题可以这么解决：

```
for_each(v.begin(), v.end(), _1 = true);
```

C++ 11 标准放出，骂的人比捧的人要多。一是认为原有的 C++ 老标准已经满足需要，二是认为新标准加入的新功能让初学者更不能适应。这两种意见有道理。

51CTO 编辑认为，C++ 11 能把 C++ 这款“老”编程语言带到一个新的高度。■

细数Java十宗罪

好吧，我知道看这标题很多人就忍不住要拍砖或表示不赞同了，我都接受。

我在遇到问题时，找一些搞 Java 朋友求助，有能解决的，我想说的是不能解决的情况下，他们大多会说："你怎么能这么弄呢？这么搞是不行的，你首先在思路上就错误了，我们从来就不这么做。"，比如"我们一般很少用存储过程，你用这么多存储过程，我建议你使用 hibernate 代码实现你的业务，而不是用存储过程"，这个问题是在我遇到用 hibernate 调存储过程时发生一些状况后，我一个搞 Java 的朋友也无法解决时，他如此说道。

Java 开发人员还喜欢说：这个不应该由 JDK 或某某框架完成，而应该是由勇敢、勤劳、智慧的您来编写实现的。

纠正

非常抱歉，第 1 点关于 "abc"=="abc" 返回 false 是我搞错了，在 java 里是返回 true 的，我没有经过实验就这么说了，非常抱歉。之所以会这么说，是因为不记得是哪一种，因为我使用了 == 进行了字符串的比较，结果并没有返回我相要的结果，然后请教老人，老人们就训诫道：最好使用 equals 进行比较，而不要使用 ==，故有此感慨。

以下都是我在开发 Java 项目时，遇到的一些问题，可能也有写得不对的地方，望指正。遇到很多问题，就随便列举了几条。

Top 1："abc"=="abc" 返回的结果是 False

很多初学 Java 的都要在这个问题很浪费很多时间，因为会非常自信的以为这里会返回 true，根本不会想到原来问题出在这里。网上看到有贴子讨论过这个问题，说什么 Java 是纯面向对象的语言，== 操作符是比较地址什么的，而 "abc" 是引用，所以不能使用 == 操作符进行比较，而应该使用 Equals 方法来进行比较，会犯这种错误的，多半是因为自己基础不牢，却还来说 Java 的不是。我看到一个人说的一个例子，很能表示我的感想，是这么说的：有一天我去到一个餐厅，因为餐厅门口有一滩水，导致路面很滑，我不小心摔倒了，于是我找来餐厅经理，追究其责任，结果餐厅经理告诉我，这是因为我自己走路的姿势不对，所以滑倒了，与餐厅无关。

Top 2：为什么没有 get;set; 属性，而使用 getXX();setXX() 方法来代替，反射不要成本？

我所了解到的 Struts\Spring\Hibernate 里都大量用到这种方式，比如 Struts 中的 VO 对象，里面若有个 getUsername()，在页面里可以用来取值，这中间我想应该是用反射来找到 getUsername() 方法，再取得其值的吧，同样的 Spring\Hibernate 中也大量用到这种方式，我想问，反射不是说很低效的吗？

Top 3：做个自定义标签还要自己写个 TLD 配置文件

有时在看一些 Java 的相关书籍上面提到

零配置时，我就觉得非常可笑，不知道所谓零配置的精神到底体现在哪里？也许写个配置文件也并没有那么难，但关键问题在于，从技术上来讲这个 TLD 明显是可以不要的啊，TLD 里面主要就描述了这个 tag 叫什么名字有哪些属性，分别是什么类型，这些信息完全可以在类里面表示，只要是实现 Tag 接口的类就被自动识别为自定义标签这样不好吗？通过识别类里有哪些 getXX();setXX();（更好的方案是有 get;set; 属性）来确定这个自定义标签有哪些属性，又分别是什么类型这样不好吗？为什么还要多此一举搞个 TLD 配置文件？

还有个问题是，如果我写了个 MyTag 的类，继承了某个自定义标签类，我还要为 MyTag 写个相应的 TLD 配置，我不知道将来还有没有其他开发人员会继承我的 MyTag，也许就算我热烈欢迎别人来继承我的 MyTag，但当别人看到我 MyTag 里近三四十个 getXX();setXX(); 之后，想到要为其写上相应的 TLD 配置文件而望而生怯吧。

Top 4：数据访问本来简单却弄得复杂

简单来就，就是执行 SQL 语句，复杂一点，就再加上实体映射，所有框架解决的问题，首先就是要易于使用，在使用过 Hibernate 之后，我感觉它太复杂了，我在 .NET 里有一个数据访问层，只需要在指定的配置文件中配置连接字符串，就可以在程序中的任何地方调用 DbHelper.Execute(sql)、DbHelper.ExecuteDataSet(sql)、DbHelper.ExecuteDataTable(sql) 了，使用起来非常简单，当然也有实体映射，DbHelper.save(entity)、DbHelper.delete(entity or key)、DbHelper.select（条件）这一组方法就可以操作实体对象，Select 返回的是实体列表，实体通过元属性设置其关联的表和字段，这中间除了连接字符串之外，是没有任何其它配置文件的。相比之下，Hibernate 咋需要这么多配置文件？我知道 Hibernate 也可以配置注解，就不需了 HBM 配置文件了，但即使这样，据我了解依旧还是需要很多除连接字符串之外的其它的相关配置文件。

Top 5：Hibernate 很难用

到底是我不会用，还是它真的就是这样的，Hibernate 对于存储过程的支持，实在让我抓狂，居然不支持存储过程，在网上寻找 Hibernate 调用存储过程，得到的答案多数就是越过 Hibernate，而仅仅从 Hibernate 中取得一个 Connection，再使用 JDBC 的方式调用存储过程，这样做存在一个问题，事务不能得到控制了，由于我还比较水，Hibernate 的事务控制又是暗箱操作的，好像是只要在 Service 层中写的业务代码就都在一个事务中，所以我无法让我的存储过程调用和 Hibernate 业务代码串在一个事务当中，而很多情况下，我是想要让它们一个失败就全部失败的。

除此之外，也有不越过 Hibernate 而调用存储过程的办法，有两个，也是要写配置文件，一个是必须要有返回的结果集，我就很纳闷，为什么一定要有结果集，我的很多存储过程就只是处理一些数据，不需要返回结果集的，最难受的是 Oracle 的存储过程其实不支持返回结果集，必须使用一种变态的游标方式返回，这么做我会感觉到极其反胃。另一个办法是通过修改实体在 Insert\Update\Delete 时的默认行为，比如我在 Insert 一个员工时，本来应该是执行 SQL 语句 insert into employee values (?,?,?,?) 的，我可以通过配置文件

细数 Java 十宗罪 III

修改这个默认行为，改成{call myproc(?,?,?,?)}，这种方式显然也不是我想要的，我只想调用一个存储过程，执行一个业务的处理。以上两种方式是会被暗箱操作的事务所管理的，但并不能满足我的需求，我要怎么办？

Top 6：数据访问的结果集对象 ResultSet、RowSet、CachedRowSet 等没得到广泛应用

各个框架更多的是倾向于支持实体列表，这么做导致出现一个问题，那就是我只能返回已知结构的结果集，若想要临时返回个东西还必须要在实体中添加相应的属性 getXX();setXX(); 方法，比如在 Hibernate 中，要访问员工表，员工表中本来只有部门 ID，没有部门名称，你想要有部门名称，就必须在员工实体中添加一个 deptName 的属性，要所有的结果都是已知结构的，这样很痛苦，如果不返回到实体列表中，也可以返回到 ArrayList 中，但这样的数据没有列名称，不明白为什么不直接查询到 ResultSet 中，然后让更多的框架支持 ResultSet，比如 Struts，在写页面使用 Struts 标签时，可以像操作实体列表一样操作 ResultSet。

...，（还是本来就支持，只是我不会？那就不好意思啦！）只是希望让更多的框架支持未知结构的结果集，让程序员事先设计好结果集的结构是很累人的，就算是代码生成，也只能生成数据库里的每一张表对应的实体，但往往我们需要 select unkownSchema from myTable 得到未知结构的结果集，并不是每次都 Select *。

Top 7：再说 ResultSet

之所以不直接用这个，而使用实体列表来代替，我想是不是也间接的说明了，ResultSet 这个类不方便使用，.NET 中的 DataSet 和 DataTable 就得到大量使用，因为它们方便好用实用。可能最大差别的地方就在于，DataSet 是断开式的存在于内存中的微型数据库，而 ResultSet 只是连接式的数据库读取器，相当于 .NET 中的 DataReader，必须保持连接才能读数据，我知道有 CachedRowSet 可以断开式的存储数据在内存中，好吧，这个就不是问题了。但另一个问题在滋扰着我，做为存储结果集的容器，提供给我们操作这个结果集的方法太少了，甚至取得该结果集的总行数的方法，我们都需要开动小脑筋，这么写：rs.last(); int count = rs.getRow(); rs.first() 负责的话，它需要至少三句代码才能取到总行数。也许这只是小问题，这个或许应该由勇敢、勤劳、智慧的我们来实现。

Top 8：在我看来，Struts 最大的意义在于，它使得每个 JSP 页面都有了一个与之对应的 Java 类的方法，也就是那个 Action 方法

你一定会跟我说，Struts 的功能并不只如此，但我说，我见过的很多（小公司）的项目，Struts 的意义就只是这样，我想像在我们国家，还有成千上万使用 Java 技术的公司，Struts 对于他们的意义，也就是让 JSP 有了后台代码。如果仅仅只是如此，为何不由官方提供，直接让 JDK 支持，让 Struts 的先进来弥补 JDK 的落后吗？只会欲盖弥彰。

或者你会说，即便 Struts 就是提供了让每个 JSP 页面都有一个与之对应的 Action 方法，这也非常伟大了，做到这一点，已经彻底改变了人们开发 Web 项目的方式，由原来的业务代码和页面混在一起，变成解偶分离，非常成功了。我想说，不要拿你十年前的荣耀到今天再来说了，已经 Out 的不行了。（剩下两宗罪，请访问原文：

http://developer.51cto.com/art/201106/269817.htm ■

编者按

覆盖金融，电信，政府，医疗，能源，公共事业，零售，物流等行业对大数据存储，挖掘均有巨大需求，本次论坛集各家所长，共同解决一个问题：面对海量数据，你准备好了吗？

大数据时代已来临，你准备好了吗？

从几拍字节的数据仓库到社交媒体数据，从基于云计算的应用程序到传感器和移动设备，从电子商务处理到地理空间信息，海量数据的时代已经来临。在已经到来的大数据量时代，数据存储发生了什么变化吗？是的，发生了巨大的变化，存储形式仿佛转了一个圈，又回到了文件式存储。据统计，包括视频、音频、图片、微博等在内的非结构化数据将占企业数据的 80% 左右，到 2012 年数据存储基本会是以文件形式存储。

在这个大时代来临之时，您是否还记得 1TB 的数据仓库被视为大储量的年代？如今，您只需要付出不到 100 美元就可以从当地零售商处购买到存储量为 1TB 的存储设备，而许多数据仓库的存储量已经超过了拍字节（PB）。

不过持续增长的数据量仅仅是海量数据的一半构成内容，海量数据同时带来了数据的多样性，复杂性以及速率的大规模增长。这种变化具有破坏力吗？是的，它具有破坏力，你做好准备迎战它，击败它了吗？这是一次商机吗？是的，这是一次商机，那么你做好准备去利用它了吗？该如何击败，该如何利用？答案只有一个：数据挖掘，挖掘出商机无限，挖掘出潜在信息。

在大众点评网 CEO 张涛看来，数据挖掘是一家互联网公司必不可少的。实际上，不只是互联网公司，数据挖掘对于任意一家公司都是必不可少的。

数据挖掘到底能做什么？

数据挖掘能做以下七种不同事情（分析方法）：

分类（Classification）

估值（Estimation）

预言（Prediction）

相关性分组或关联规则（Affinity grouping or association rules）

聚集（Clustering）

论述和可视化（Description and Visualization）

复杂数据类型挖掘 (Text, Web , 图形图像，视频，音频等）

数据挖掘中的算法

“数据挖掘算法”是创建数据挖掘模型的机制。为了创建模型，算法将首先分析一组数据并查找特定模式和趋势。算法使用此分析的结果来定义挖掘模型的参数。然后，这些参数应用于整个数据集，以便提取可行模式和详细统计信息。

数据挖掘，越来越多的体现在企业的数据报表上，也会为我们带来越来越明显的效益。所以，您做好准备了吗？做好准备迎接新的时代，利用多样化数据的准备了吗？

客户的一次疏忽，DBA的一次噩梦

编者按

你是否有过如下的经历呢？是否半夜被老板挖起来，只为了客户的一个小小的误操作？

今晚接到老大的电话，泰国的客户不小心删除了一些表的数据，现在非常着急，需要恢复数据。其实 DBA 做的数据库备份，很大程度是用于数据库 crash 掉的时候，恢复数据，而不是三天两头的因为客户误删了数据，而去做恢复。

看了客户的邮件，是有 2 个表的数据被误删除或者误插入或者误更新了。总之，操作过一大通，希望恢复到当天下午 15：30 的数据。上数据库去查了一下，用备份来恢复，似乎时间不够，尝试用户 flashback query，发现已经回不去了：

```
SQL> SQL> SQL> SELECT count(*) from hr_ttm.TA_ABSDOCS
  2  AS OF TIMESTAMP TO_TIMESTAMP('2011-06-09 15:29:00','YYYY-MM-DD HH24:MI:SS');
SELECT count(*) from hr_ttm.TA_ABSDOCS
                         *
ERROR at line 1:
ORA-01555: snapshot too old: rollback segment number 1 with name "_SYSSMU1$"
too small
```

其他也没有更快的方法了，于是当下决定用 logmnr 挖数据，

由于数据库原来就没有配置 utl_file_dir，因此还需要重启数据库使得该参数生效。一路做下来，大致算顺利，不过也遇到了不少小插曲。下面就是恢复的步骤：

一、备份原表

```
create table hr_ttm.TA_ABSDOCS_20110610_0010 as SELECT * from hr_ttm.TA_ABSDOCS;
create table hr_ttm.TA_ABSDOC_20110610_0010 as SELECT * from hr_ttm.TA_ABSDOC;
```

二，根据客户要求，建立新用户，将恢复的数据导入到这 2 个表中：

```
create user hr_ttm2 identified by hr_ttm2 default tablespace MSG_DATA;
grant connect,resource,dba to hr_ttm2;
```

三、把原表数据备份到新用户下，用于做回滚

```
create table hr_ttm2 TA_ABSDOCS as SELECT * from hr_ttm.TA_ABSDOCS
create table hr_ttm2.TA_ABSDOC as SELECT * from hr_ttm.TA_ABSDOC
```

四、修改参数，用于挖日志，重启数据库

```
alter system set utl_file_dir='/prodlog/logmnr' scope=spfile;
```

五、生成数据字典

```
exec dbms_logmnr_d.build('dictionary.ora','/prodlog/logmnr');
```

做这一步之前注意需要修改 LD_LIBRARY_PATH 和 LIBPATH，使得 lib 的变量在 lib32 前面。（本文未完，更多内容请访问下面的原文页面：http://database.51cto.com/art/201106/271847.htm ■

编者按

本文是从 Vic Cherubini 的博文《What PHP Needs to Change》翻译而来，作者从自身对 PHP 语言的理解和喜爱为出发点，谈了对 PHP7 的展望，文章列举了作者认为 PHP 需要改变的地方，并祝 PHP 未来更美好！

PHP 7展望：PHP需要改变什么

程序员（一般）定义好的编程语言是他们知道最好或最经常使用。 我是一个 PHP 程序员，约翰是一个 JavaScript 程序员，DHH 是一个 Ruby 程序员，施瓦茨和兰德尔是一个 Perl 程序员。 每个不同程序员看待问题都是不一样的，但是总有一两个会被共同认为是最好的。

每当听到 PHP 被不尊重的时候，作为一个 PHP 开发人员，感觉很受伤。

虽然我们必须使用 PHP 为一个用于 Web 应用程序的语言，但它是一个烂语言，其使用不应予以鼓励，或支持超出了必要。

哎哟，听说我日常使用的语言，被给予很烂的评价时候，有人可能会说你不应该管其他人对你选择的语言的评价，只要自己认可就行。我想我真的不应该吗，而且该声明暗示的是，不仅是语言很烂，开发商烂，而且我们 PHP 开发者也烂。

那么，究竟应该怎么做才关闭他人的大嘴巴呢？我们能否使 PHP 语言受到尊重？我们能否让 PHP 开发人员受到尊重？让我们来分析某些方面我们可以做到这一点。

打破一切

PHP7 要打破一切。 PHP 开发人员应该接受打破版本之间向下兼容的定律。只要不允许大量的向后兼容，PHP7 将是一个高度尊重的语言。

◆ 创建一个具体的核心语言 – 删除所有库方法，并保持在对象集中的核心方法。 您应该能够编写无需任何外部库或扩展 PHP7 和对基本输入 / 输出，字符串处理和数学一个很好的完整的语言。库以外的任何应该通过批准扩展。

◆ 一切都当作一个对象 – 以从 Ruby，Smalltalk 和（主要）的 Java 对象，并把它一切当作对象。

整数是对象，字符串是对象，他们每个人都可以操作的方法， 我不相信 PHP 需要的 Ruby 和 Smalltalk 在对象之间传递彼此讯息的观念，而调用对象的方法才是最好的。

◆ 一致的命名方法和类 – 由于 PHP 的最大的抱怨之一是不断要检查，(needle,haystack) 或 (haystack, needle)， 或 some_function()， 或 function_some()，或 someFunction()，一个一致的格式需要制定。

◆ 让事情严格 – 尝试传递到一个方法浮动字符串？这是一个警告。

◆ 一切是 Unicode – 在 PHP6 中的所有字符串都是 Unicode，这很好，我主张 PHP7 也应该保持。

◆ 中央启动点 – 创建一个主类或初始化，所有代码执行源于此。

PHP 7 展望：PHP 需要改变什么 II

◆ 清理 C 代码 – 我不是一个 C 的专家，但如果你比较了解 Ruby 的 C 代码到 PHP 的 C 代码，可以很容易地了解了 PHP 与 Ruby 的内部。我非常熟悉 PHP，所以我自己的写扩展更容易。

◆ 摆脱 eval() – eval() 是邪恶的。 如果你正在使用它，那么这是一个错的主意：这将打破 PHPUnit，抛弃它从现在开始。

◆ 支持操作符重载 – 因为一切都是对象，开发者只需掌握操作对象的方法即可。

◆ 允许的方法签名 – 允许真正的方法签名，所以程序员可以有不同的参数列表或返回类型的同名方法。

```
class A {
  public int function doSomething(int $a, float $b) {
    // Same as $a->*($b->to_int());
    int $c = $a * $b->to_int();
    return $c;
  }

  public float function doSomething(int $a, float $b, float $c) {
    // Same as calling $a->*($b->*($c)); since * is a method on each object $a and $b.
    float $d = $a * $b * $c;
    return $d;
  }
}
```

◆ 建立一个 PHP 虚拟机 (PVM) – 我不能完全肯定这是可能的，因为我不是一个语言设计师，但它会是不错的一个 PHP 虚拟机。 它可以执行 PHP 字节码，并允许一个明确的堆和堆栈。

◆ 删除 copy-on-write (COW) –COW 是一个相当陌生的概念，，如果你不知道它的存在，以新的开发，就可能导致问题。

◆ PHP 官方发布规格 – 类似于 W3C HTML5 规范，PHP 的规格将允许开发人员实现自己的 PHP 版本，并确保有具体的例子来编译。

尊重语言

应努力使语言得到尊重。 我们应尽量招募开发者做出具有非常强大功能的 PHP7。 我们应该释放巨大的代码的安全，易于阅读，并教新的开发者以正确的方式编程。

我不知道 Ruby 的疯狂，但我尊重 Ruby 语言。 我看到它的力量，我看在 Ruby on Rails 是一个非常好的框架，它只是我不是喜欢的。如果我们能获得具有很好威望的开发商认可或尊重，PHP 就会走的更远。

尊敬开发人员

从上文的理解，我们需要一个非常德高望重的 PHP 开发核心团队。他们发布代码，举办讲座，向人们展示了“正确的方式”做事，这支团队将很快的受到不少开发商的尊重。

结论

我兴奋 PHP 的未来。 我很怀疑我的想法会被执行，但我真的相信他们会帮助整个社会。 我感到非常兴奋，当世界第二大的网站宣布，他们正在帮助 PHP 建立伟大的社区。

PHP 是不会消失，他会越来越好，只要我们努力提高它，改善它，终有一天 PHP 会成为受开发者普遍尊重的语言。

最后，我不是一个语言设计师。 我的想法可能会被认为完全胡说。 如果我不正确的地方，请礼貌地让我知道，我会很高兴地谈论它。 让我们共同努力，使 PHP 成为备受尊重，功能强大，快速，高效的语言。

本文未完，更多内容请访问下面原文页面：

http://developer.51cto.com/art/201107/272690.htm ■

Google主页实现月食实时记录揭秘

谷歌近来一直在尝试将各种新概念加入到其主页上的主题栏当中，而谷歌如何的主题栏中实现各类创意也早已不是秘密。不过今天，他们抛出了更为天马行空的创意：在主页上实时记录整个月食过程。

近十年以来持续时间最长的一次月食当初也是由谷歌在 youtube 上实时放送的，这一事件同时登上了谷歌的主题栏。现在就让我们回顾一下整套实时放送是如何实现的吧。

当时的主题栏月食图案在概念阐释方面较为直观，因为它是 CSS 3 脚本与 JavaScript 脚本协同处理的结果。

1. 背景图片

如大家所见，这是一幅静态图像，中央部分为月亮留出了透明的圆形空缺，并在月食过程中即时加载不同的效果。

2. 动态月球影像

与之前利用 CSS 脚本的做法不同，这一次主题栏会采取动态影像的处理方式。今天我们所看到的实时月球影像，实际上是一系列拍摄间隔较短的实景照片的有序排列。

这些图片是由谷歌拍摄并实时添加到资源库当中的，因此无论大家何时刷新页面，主题栏中出现的总是正确的当前月食情况。

而下面这幅细长的图像则显示了月食发生较长时间之后的情景。很明显月球图案的采集数量增加了，这意味着当前的最新情景处于最右侧图片的状态，而动态效果则也是由满月向最右侧的状况转变。

调用图像若仍然用 CSS 3 脚本动画实现，无疑随着动画长度的增加，文件也会变得相当庞大。

而在月食发生的一百分钟里，谷歌需要保证按时上传当前最新的月球图片。在此前的 Martha Graham(美国现代舞表演创始人) 主题中，所有的舞蹈动作都被描绘在同一幅图片中。但这一次，月食图片需要以动态方式添加及创建。

全部月球实景图都调用自下列网址

URL ：http://www.google.com/logos/2011/eclipse/strip.jpg?cache=1308162471 且此网址并非固定不变的。

3. 动画：

这里我们要说到月食动画。当 google.com 页面进行载入时，以上所列的图片也将同时在浏览器中进入缓冲状态。当整体缓冲动作结束时，动画随即开始播放。当图片按脚本的安排由左侧向右侧逐幅显示时，动态动画也就呈现出来了。

默认的月球图片 (即左侧第一幅) 会先于其它一系列图片被载入。因此在全部图片都缓冲结束之前，我们不会看到其它帧中的内容。

4. 编码

其实这里并不涉及太多的编码工作，除了每个坐标点需要载入的 CSS 3 脚本。

本文更多详细内容请查看：
http://developer.51cto.com/art/201106/269588.htm ■

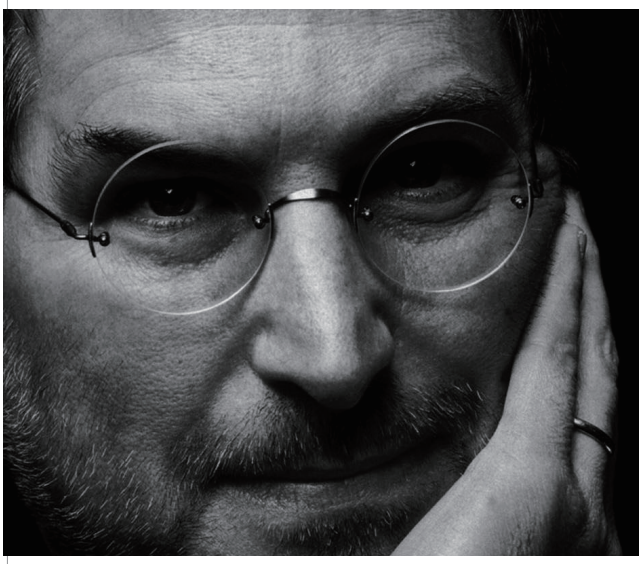
编者按

iOS 5.0 这个新版本定于 2011 年秋天上市。近日史蒂夫　乔布斯着重介绍了苹果 iOS 5 这个新版本的 10 大特性，不妨一睹为快。

乔布斯透漏iOS 5.0十大新特性

【51CTO 译文】终于快要等来新的 iOS 了。苹果公司宣布了 iOS 5.0 为 iPad、iPhone 和 iPod 添加的新特性超过了 200 项，这个新版本定于 2011 年秋天上市。近日史蒂夫　乔布斯着重介绍了苹果 iOS 5 这个新版本的 10 大特性，不妨一睹为快。

向史蒂夫 · 乔布斯致敬！



这是个时时发生奇迹的世界。史蒂夫乔布斯击破了身患重病、离任苹果的种种传闻，登台亮相向世人发布新的苹果 iOS 5.0，又一个奇迹随之诞生。

我很高兴史蒂夫乔布斯身体无恙，祈祷上帝让他永远健康。我真的很欣赏这个人，正是他为信息技术界带来了一场全新的革命。

iOS 5.0 平台确实让人叹为观止，它拥有诸多新特性，比如通知中心、Newsstand、iMessage、改进的 Reminders、Twitter 功能 (Twittering Capability)、新的 Camera 应用程序、Photos、更快的 Safari 浏览器、WiFi 同步和其他众多特性。我们不妨逐一了解这些特性。

通知中心

新的 iOS 5.0 包括了就在屏幕上面的一个通知中心，让用户可以查看各种各样的通知，比如未读邮件、短信、日历事件、提醒、天气报告以及其他许多通知。这个新的通知中心特性其灵感来自**未读邮件、短信、日历事件、提醒、天气报告以及其他许多通知。**

Newsstand

Newsstand 让你可以使用一个名为 Newsstand 的文件夹，添加、删除或管理你拥有的所有订购报刊和杂志。有了这个新的应用程序，用户可以快速方便地访问所有报刊杂志。Newsstand 最好的地方在于，它在后台运行，一旦你订购的东西有新的内容，它就会更新。

iMessage

这可是我长期以来梦寐以求的特性。这回它终于出现在新版本中。大家知道，有了这个新的 iMessage 应用程序，就可以通过无线方式，实际

乔布斯透漏 iOS 5.0 十大新特性 II

发送消息、电子邮件、照片、位置信息、视频，甚至与朋友聊天，甚至不用付一分钱。

改进的 Reminders

顾名思义，Reminders(提醒) 让你可以创建待办事项，那样就能记住日常生活中的重要事务。更为重要的是，由于集成了 iCloud，更新内容可以在你的所有 i 设备 (通常指苹果公司的 iPad、iPhone、iPod 和 iPod Touch) 上都能自动实现同步。新的 Reminders 应用程序让你可以记住朋友的生日。

Twitter 功能

不需要我再来描述 Twitter 是什么东东。我们都一直在经常发送大量 Twitter 消息，也很了解 Twitter 的真正魅力。你也知道，美军准备抓捕本拉登时，其藏身处附近的一个人在 Twitter 上发布了关于这次行动的消息，说是听到了枪战和直升机的响声。

新的 Camera 应用程序

装有 Camera 应用程序的 iPhone 或 iPad 要是无法让你捕捉意外的美好瞬间，那也没啥用处。大多数人希望能够捕捉到瞬间的照片。iOS 5.0 中新的 Camera 应用程序让你只要点击一下按钮，就可以捕捉到你想捕捉的任何瞬间。

Photos

新的 Photos 应用程序可以让你随意处理通过 Camera 应用程序捕捉的照片。有了这个新的应用程序，你可以修饰现有的照片、裁切照片、旋转照片，甚至还可以消除红眼效应。

更快的 Safari 浏览器

借助新的更快更好的 Safari 浏览器，你势必会获得更好的互联网浏览体验。新的 Safari 浏览器让你可以阅读文章，界面整洁，又不会因广告或其他的自定义弹出窗口而分心。

WiFi 同步

新的 iCloud 特性确实让人叹为观止；我在体验了它的真正魅力后，开始喜欢上了它。史蒂夫乔布斯在竭力借助新的 iCloud，让人们不再需要电脑。你认为他能够掀起这样一场宏大革命吗？

新的 iOS 5.0 让你可以通过无线连接，将 iPhone、iPad 或 iPod 上的内容无线同步到 Mac 机或个人电脑上。还是觉得这项新特性没啥过人之处？它有更多的本领。

有了 iCloud 和 iOS 5.0，你再也不需要用电脑来同步 iPad 或 iPhone。现在使用 iCloud，甚至就可以恢复或更新你的 iPhone。这意味着你不再需要时不时地将 iTunes 和电脑连接到 iPhone。iCloud 有助于自动更新和备份你的所有 i 设备，不需要用户的干预。

辅助功能、游戏中心及其他

iOS 5.0 新的辅助功能让你可以用 iPhone 或 iPad 来做更多的事情。借助新的辅助功能，你就可以全面利用硬件，比如闪烁或震动器，以便提醒你有人打来电话，同时不会干扰别人。

对世界各地的游戏玩家来说，新的游戏中心是一站式平台。游戏中心让你可以与其他使用 iPhone 或 iPad 的玩家联系起来，从而让你能够选择游戏对手、发布个人档案照片或处理其他的众多事情。

完整文章请点击 51CTO《乔布斯向开发者透漏 iOS 5.0 十大特性》

http://mobile.51cto.com/hot-273255.htm ■

开发"愤怒的小鸟"的Wax框架

我们都知道 Objective-C 和 Cocoa 语言可以开发 iOS 应用，但是一年前，苹果决定在 iOS 系统上使用 Lua 语言。Wax 框架的想法很简单：凡是 Objective-C 能做的，Lua 也能做！

2011 年 6 月的编程语言排行榜 Lua 语言一军突起，一举进入前十名。这与一年前苹果决定在 iOS 系统上使用 Lua 语言密不可分。但是，你了解如何用 Lua 语言在 iOS 上开发应用吗？

51CTO 将向各位介绍 Lua 语言的 iOS 应用开发框架——Wax，其中在 iOS 平台上无比火爆的《愤怒的小鸟》就是由 Lua 语言用 Wax 开发的。

全文共分两部分，第一部分 51CTO 将带您深入探讨 Wax 具有的一些好处，同时演示把 Lua 与 Xcode 4 和 iOS 软件开发工具包（SDK）集成起来必不可少的实际步骤。

第二部分 51CTO 将逐步介绍如何用 Wax 构建一个简单的应用程序，显示 Twitter 上的当前趋势话题列表，可以用按钮来更新内容。

Wax 是什么？

Wax for iPhone 这种框架在开发时，旨在把 Lua 脚本语言和原生 Objective-C 应用编程接口 (API) 结合起来。这意味着，你可以从 Lua 里面，使用任何和全部的 Objective-C 类及框架。

从技术上来讲，Wax 结合了 Objective-C 类和原生 C 代码。Lua 语言嵌入了 C 语言，然后 Objective-C 类并入到其中。

为什么使用 Wax?

Wax 是免费的、开源的。与其他一些基于 Lua 的移动开发解决方案不同，Wax 是个开源框架，只需要你花一点点时间就可以上手，不需要花钱。不喜欢 Wax 的工作方式，或者发现实施方面的缺陷？源代码可免费获取，你总是可以改动源代码，以满足自己的需要。

可以利用原生 API

这意味着，为教 Objective-C 而编写的教程很容易由 Lua for Wax 来改动和编写。

这还意味着，你的应用程序在外观感觉上总是如同原生应用程序，不过又得到了用 Lua 这种高效脚本语言编写代码可以节省时间的好处。

可以使用 Xcode。这意味着，模拟器和设备部署都轻而易举，不会轻易与未来 iOS 版本决裂。

可以利用所有现有的 Objective-C 库。如果你有一个 Objective-C 类是以前编写的，不需要改动，就可以将它用在 Lua 中——只要把它放入到 Xcode。Three20 之类的库也是一样。

只要按照正常指令来添加库，就可以使用 Lua 代码访问它们。

可以利用 Wax Lua 模块。Wax 有几个内置的 Lua 模块，使得异步 HTTP 请求和 JavaScript 对象标注 (JSON) 创建 / 解析极其容易而快速（因为模块是用 C 编写的）。

没必要管理内存。不再需要操心内存分配

开发 " 愤怒的小鸟 " 的 Wax 框架 II

之类的事务。Wax 为你处理这一切。

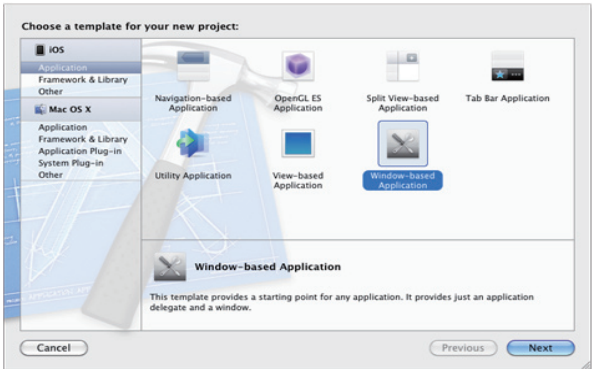
Lua 类型自动转换成对应的 Objective-C 类型，反之亦然。这意味着，如果你调用了需要 NSString 和 NSInteger 的某个方法，但传送了 Lua 字符串和 Lua 整数，Wax 会为你搞定转换工作。这种转换功能强大，甚至可以处理复杂的 Objective-C 特性，比如选择器。你可以利用所有上述特性。不需要精挑细选。你获得所有特性！

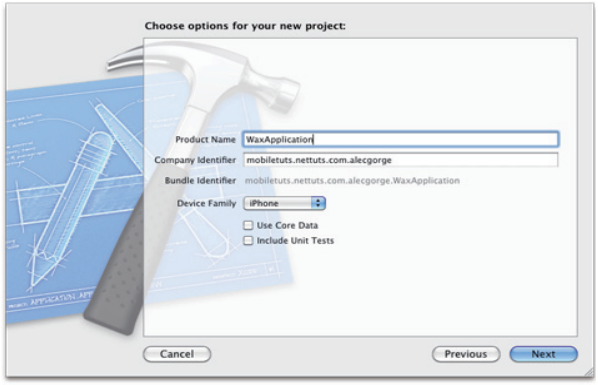
OK，实在太棒了！我该如何安装 Wax?

首先你需要 Xcode 和 iPhone SDK。要是你还没有这些东西，赶紧弄一份！

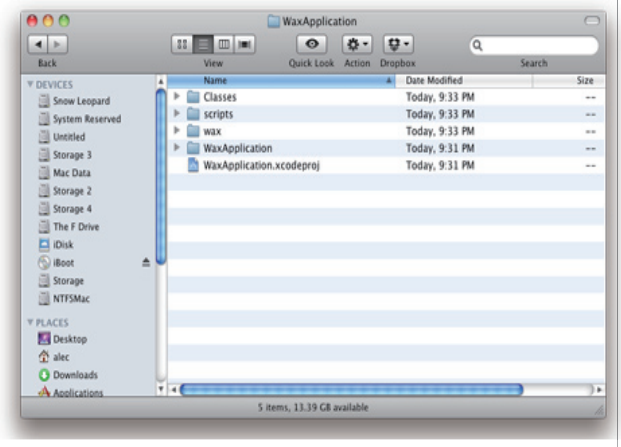
用 Xcode 创建项目

我们先创建一个新的“基于 Windows 的项目”，名为“WaxApplication”。别忘了把设备设置成 iPhone：





通过 Finder 浏览到你保存该项目的文件夹。创建三个新的文件夹：wax、scripts 和 Classes。你的文件夹看起来应该像这样：

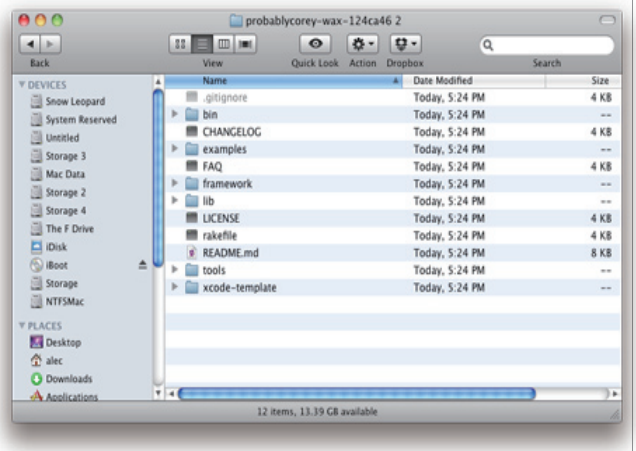


设置 Wax(第一部分，处理文件)

首先，下载源代码的压缩包。Wax 放在 GitHub 上 (https://github.com/probablycorey/wax)，那样下载源代码就很容易。从这里下载压缩包。

现在，解压缩刚下载的文件。浏览到刚解压缩的文件夹。它会有“probablycorey-wax-124ca46”之类的名称。

你的屏幕现在看起来应该像这样：



本文未完，查看更多内容请访问下面原文内容：

http://mobile.51cto.com/hot-269986.htm

http://mobile.51cto.com/hot-270208.htm ■

编者按

2009 年 Oracle 收购 Sun, 当时就有人猜测谷歌也可能是竞购的买家 . 谷歌有意收购 Sun 是为了实现提升自有软件平台的野心 . 假如当初谷歌得到 Java, 谷歌会为其提供特殊的发展空间 . 而最终 , 谷歌却错过了 Java.

谷歌史上最大失误：错过Java

早在 2009 年，Oracle 公司就斥资 74 亿美元收购了在 Unix 服务器硬件领域光彩夺目的制造企业 Sun。许多人当时都在猜测谷歌是否也是此次竞购的买家之一。谷歌的资金储备无疑是雄厚的，不过真正的问题在于收购的理由：许多分析人士都认为 Sun 公司的黄金时期已成往事，谷歌并没有充分的理由将其招至麾下。

当然，现在 Oracle 公司已经拥有了与 Java 相关的一切专利及版权 (Java 这项技术最初正是由 Sun 公司所创立的)，而谷歌也极有可能为自己的犹豫付出代价：尽管并未取得授权，谷歌的 Android 系统中仍然用到了 Java 技术。根据目前的情况来看，Oracle 公司的赢面相当大，无论是即时生效的罚款裁决、还是根据 Android 平台的实际盈利按百分比计费，当初付出的 74 亿美元收购金额中的很大一部分恐怕都要由谷歌来承担了。

Oracle 公司从 Android 盈利中强插一脚的现状并非谷歌应当收购 Sun 公司的惟一理由。换句话说，这实际上应该被称为战略失误。这种失误不仅使谷歌费尽心力所打造的标准化软件开发平台在竞争前景方面蒙上了一层阴影，更动摇了近年来苹果及 Anroid 得以借势腾飞的这场设备革命的根基。

为了公平起见，51CTO 认为历史上由并购活动所造成的巨大失误屡见不鲜。网络电视并购事件最终带给我们的是微软 IP 电视，即 Mediaroom 的领先地位。当然，这里所谓的“领先”是指我们等了八年以上才见到真正可行的 IP 电视平台搭建技术，只是因为这套技术要保证能被微软的其它电视及视频相关技术所兼容。说到这，合并史上的典型失败案例该登场了：微软斥资 5 亿美元收购 Danger 公司。这家公司的创始人之一安迪 · 鲁宾 (Andy Rubin) 如今在为谷歌工作，而他所负责的正是 Android 项目。Danger 公司的资源一直集中在前途未卜的 KIN 上，这是一款曾经对微软 Windows Phone 7 造成过相当威胁的设备。不过平心而论，并购也并不总是带来灾难。谷歌对 YouTube 耗资 10 亿美元的收购就是标准的成功范例。当然，带宽成本的局限使得谷歌很难从 YouTube 项目本身上盈利，但 YouTube 作为互联网视频提供者中的领军品牌，肯定能够在其它诸多方面向谷歌回馈丰厚的利润。而且连 IP 电视及智能手机也将支持访问 YouTube 当作一项基本功能。

谷歌当初有意收购 Sun 公司在很大程度上是为了实现其提升自有软件平台的野心。Solaris 系统对谷歌意义有限，而且有可能会被那些关注客户硬件情况的买家 (Oracle 公司就是主要候选

谷歌史上最大失误：错过 Java II

人之一）单独拍下。Sun 公司的软件才是重中之重，他们自主研发或是拥有的软件能够为任何一家企业在 .NET 或是 Objective-C/Cocoa 方面提供进行开发所必备的强大 API；不仅如此，这些软件同时提供了强大的办公套件，能够以特殊的方式对自己的在线文字处理工具进行补充及增强。这两个方面对微软来说绝对是实实在在的威胁，因为它们所针对的正是微软公司立足的根本（即 Windows 系统平台及 Office 办公软件）。

假如当初是谷歌拍到了 Java，他们无疑会为其提供特殊的发展空间，类似一款 API，旨在帮助这项已经相当流行的技术进一步普及。大多数刚刚走出校门的程序员对这款平台非常熟悉，而且 Java 在包括电视及移动设备在内的诸多领域都具备深厚的覆盖基础。当然，谷歌同时也不会停止对其它开发技术的支持或是放缓将应用程序客户端推向网络接口的脚步。无论如何，即使在客户端领域，本地应用程序也还有不少潜力可供挖掘。Andy Rubin 以及 Android 团队很清楚这一点，这也是他们在开发技术方面选择了 Java，而不是像 Palm 在 Web OS 项目中那样选择 HTML/CSS/JavaScript 之类的原因。

这样一种定位如今可以说恰逢其时。微软眼下正积极筹备自己的手机及平板设备，而确定要应用在这些设备上的未来计算平台都不是微软自家出品。

进一步来说，微软正为其用户界面战略酝酿一套大规模的改革方案，这其实给那些与微软公司向来保持亲密合作关系的开发商们带来了不小的压力。微软最近发布的 Windows 8 新基础概念吸引了很多人。开始屏幕将支持那些由 HTML、CSS 以及 JavaScript 所编写的应用程序，这不仅颇具趣味性、更是一种重大进步。而 Windows 开发商们则注意到，.NET 似乎成了被遗弃的孤儿。那些由 .NET 所开发的应用程序在全新的 Windows 8 桌面系统上将无法继续保持任何具有倾向性的特权。更让人费解的是，微软拒绝对上述情况做出明确回应，无论是证实还是否认。

也许，正如许多人所主张的，这完全是对谣传的一种肆意夸大。然而，微软公司在与开发商的交流方面一直做得很到位，像这样不同于以往的“失误”反而进一步加剧了 .NET 将被打入冷宫这类传言的可信程度。正如 Tim Anderson 最近在 The Register 网站上所指出，“在外界看来，好像微软的服务器及工具部门在向一个方向努力，而 Windows 团队则将目标定在了另一个方向。”作为一名曾在微软就职过的前员工，我得说公司内部不同产品的开发团队之间在配合方面的表现糟糕到不能解释。由此看来，.NET 开发社区中弥漫着的悲观氛围似乎确实具有合理性。

微软的痛苦大概会成为谷歌的快乐。祸起萧墙的微软实际上是给竞争对手提供了一个千载难逢的机会；而抓住这个机会的谷歌打造了一个平台，一个足以当作备选方案的可靠平台。

谷歌希望自身成为应互联网时代之需所诞生的平台，这在其对浏览器的重视程度方面就得到了体现；但显然，Android 系统的成功让谷歌在成功之路上又迈进了一大步。■（完整文章请点击 51CTO《谷歌史上最大失误：错过 Java》

http://mobile.51cto.com/hot-270801.htm

中国移动OS进入“大航海”时代

7月7日，点心OS发布了与海尔合作的第一款平板电脑产品“HaiPad”。51CTO第一时间采访了点心OS的软件架构师梁泉先生，与梁泉一起聊聊“竞争产品的差异化”和“HaiPad”。

中国本土的移动操作系统发展迅速，从运营商的移动、联通分别发布了自己的“OMS”和“沃Phone”操作系统，到国内PC龙头的联想“乐OS”系列的“乐Phone”、“乐Pad”。7月7日，点心OS发布了与海尔合作的第一款平板电脑产品“HaiPad”。中国本土操作系统进入空前繁荣的“大航海”时代。51CTO第一时间采访了点心OS的软件架构师梁泉先生，与梁泉一起聊聊“业界开发者最关注的竞争产品的差异化”和“HaiPad”。

“开放”是大航海时代成功的要诀

大航海时代是欧洲由黑暗的中世纪向工业革命过度的时代，这个时代也是一个英雄辈出的时代。在中国互联网向移动互联网的时期，众多移动相关的企业把移动OS作为移动互联网这场战争的核心争夺点。点心与海尔在7月7日联合发布了“HaiPad”，中国移动OS的运营商与PC厂商竞争时代将一去不复返，软件厂商将成为这场角力的新一极。

不久前51CTO采访了人人公司副总裁吴疆先生，吴疆在采访时反复提到了“开放”这个词。在与梁泉采访的时候，梁泉在面对“业界开发者最关注的竞争产品的差异化”这个问题的时候首先提到的便是“开放”：“第一个就是开放。这是跟其他平台比较大的一个差别。在Android发展起来的过程中，开放起到了极为关键的作用，点心非常赞赏和支持这一策略。因此，在自身的技术架构上，点心做到了跟Android的完全兼容和并存，并保证通过CTS。同时，在合作方面，点心也非常开放，无论从技术的角度(比如开放部分源码，推SDK等)，还是商务合作的角度，都希望加强与更多开发者和厂商的合作交流。一起推动Android的发展。”

同时，在谈到与开发者如何合作时，梁泉表示：“点心是开放的平台，非常欢迎跟第三方开发者的合作。

点心OS自身首先保证了和Android的完整兼容性，第三方开发者可以放心的将自己开发的Android应用直接跑到点心OS上来，而不用担心有兼容问题。

同时，点心未来也会考虑提供点心SDK，一则方便开发者使用点心提供的扩展框架功能，二则鼓励开发者接入点心的云服务，共同给点心的用户带来更有价值的应用和更丰富的内容。”

中国移动 OS 进入“大航海”时代 II

点心为更多的智能终端做好准备

在最近几年里，移动通信和互联网成为当今世界发展最快、市场潜力最大、前景最诱人的两大业务，它们的增长速度都是任何预测家未曾预料到的，所以移动互联网可以预见将会创造怎样的经济神话。迄今，全球移动用户已超过 15 亿，互联网用户也已逾 7 亿。中国移动通信用户总数超过 3.6 亿，互联网用户总数则超过 1 亿。这一历史上从来没有过的高速增长现象反映了随着时代与技术的进步，人类对移动性和信息的需求急剧上升。越来越多的人希望在移动的过程中高速地接入互联网，获取急需的信息，完成想做的事情。所以，现在出现的移动与互联网相结合的趋势是历史的必然。

目前，移动互联网正逐渐渗透到人们生活、工作的各个领域，短信、铃图下载、移动音乐、手机游戏、视频应用、手机支付、位置服务等丰富多彩的移动互联网应用迅猛发展，正在深刻改变信息时代的社会生活，移动互联网经过几年的曲折前行，终于迎来了新的发展高潮。面对移动互联网的大潮，梁泉表示：“点心非常看好智能手机和平板电脑未来的大幅增长，尤其看好 Android 在这个市场中将会占据的份额。7 号首款搭载了点心 PAD 解决方案的智能手机平板电脑 HaiPad 发布了，也是为平板乃至更多的智能设备做好准备。”

总结

Android 作为一款开源产品为中国移动互联网企业的自主研发创造了可能，无论是运营商、PC 厂商，还是点心这样的软件厂商。做最贴近中国用户使用习惯的移动 OS 成为这个大航海时代的核心，“开放”、或者说与更多中国开发者进行合作更是大航海时代核心的核心。这已经成为移动互联网时代的铁律，谁能抢占这一制高点必将成功。

点心操作系统架构师——梁泉

在 7 月 16 日 51CTO 移动开发技术沙龙“Phone Club”梁泉先生将做为讲师与我们一起分享：

《如何使你的移动系统架构稳定、高效》

完整文章请点击 51CTO《点心 HaiPad 发布标志中国移动 OS 进入“大航海”时代》

http://mobile.51cto.com/news-274518.htm ■

编者按

6 月 23 日 51CTO 移动开发频道记者专访人人公司副总裁吴疆，吴疆先生将向我们介绍人人网移动端同一性服务、人人网在 SNS+LBS 上新的产品方向、盈利模式和未来的长期规划。

人人副总裁吴疆SNS+LBS=SoLoMo

SNS 社会性网络服务因 facebook 的成功而广受关注，国内 SNS 网站同样也延续了 SNS 社会性网络服务在中国的成功。4 月的 GMIC2011 大会以“SoLoMo”为关键字，使社交网络服务移动化这一概念一夜之间走进了业内人士的脑海中。6 月 23 日 51CTO 移动开发频道专访人人公司副总裁吴疆，在面对 SNS 未来如何发展时，吴疆先生是这样形容的：“SoLoMo 是一个趋势。人人网目前在做的 LBS 就是暗合 SoLoMo 这个趋势，人人网的本身是一个 Social，在移动设备上是一个 Mobile，LBS 实际上是一个 Local。人人网将引领 SNS 在移动互联网走向新的高度。”

早在 2008 年，ABI Research 的一项调研显示，以 Myspace 和 facebook 为主的 SNS 用户有 47% 同时也用移动终端登录自己的帐户，SNS 的移动化趋势不可逆转。而国内的 SNS 移动化的发展速度同样发展迅速，曾有报道索尼爱立信与开心网独家合作，其四款手机将被内置移动“开心网”软件，开心网将因此开辟 Web 之外一个更潮的生存通路，报道称“SNS 网站必然走向移动化”；而作为国内 SNS 行业的另一个巨头人人网，每天有超过 30% 的用户使用手机登录人人网。可以说，中国的 SNS 网站经过几年的发展和沉积，在 SNS 移动化的道路上取得了非常大的成绩。特别是从 2010 年开始，中国的 SNS 网站在移动互联网的各个平台上投入了巨大的人力和财力，呈现出欣欣向荣的景象。据吴疆介绍，移动互联网发展迅速，各个平台层出不穷，人人网为了能够覆盖各个平台，在移动开发的人才招聘上花了很大力气，目前人人网已经覆盖了多个移动开发平台，未来人人网会覆盖十几个、甚至几十个平台。

日益增长的社交网络服务移动化需求量

繁荣并不能掩盖问题，SNS 社会化网络服务的发展速度与用户需求增长速度显然是不成比例的。目前移动操作平台发展迅速，iOS 和 Android 两大平台的手机出货量增长迅猛，而微软和诺基亚合作将大大促进 Windows Phone 7 平台发展，Symbian、BlackBerry、webOS、MeeGo 和 MTK 等平台同样拥有自己的忠实用户。吴疆对社交网络服务移动化有着自己的认识：“社交产品需要我们不断地关注沟通联系，我们所有的产品包括网站、手机客户端都保持相同的用户体验和服务，无论你使用什么设备登陆人人网的应用我们的服务必须是一样的，具有同一性，这样人人网的产品才具有竞争力。”**移动互联网各个平台都有自己的特点。保持用户体验的同一性和最大化突出各个平台特点是人人网追求的目标。**

“SNS 要发挥每个移动操作系统的优势和

人人副总裁吴疆："SNS+LBS=SoLoMo" II

设备的优势，在两者之间 SNS 的移动开发团队一定要寻找一种平衡，也尽量把两者结合起来，我们的从业者还要有很长的路要走。”吴疆如是坦言。随后吴疆在 iOS 和 Android 两款产品上演示了人人的“九宫格”见面和“日志”界面。人人的整体风格没有变化，用户可以很自然地了解各种功能。同时，iOS 设备与 Android 设备不同的特点也放大到最大化。

如何将 SNS 与移动化紧密结合起来，这成为 SNS 开发者通往成功的一道鸿沟，人人网又是怎么完成的呢？

人人公司副总裁吴疆（左一）

SNS 未来发展：SoLoMo =SNS+LBS

当 51CTO 记者询问 SNS 网络服务如何与 LBS 结合及 SNS 在盈利模式上的探索时，吴疆向我们介绍一个新概念：SoLoMo，即：Social（社交的）、Local（本地的）、Mobile（移动的）。

吴疆认为：“LBS 重新定义了人和他发布的信息，你会发现你的信息、你的内容和你的好友会形成一个立体化的内容，而不仅仅是一个孤立的内容本身。SNS+LBS 完全可以重新定义每个人，还有他发布的内容价值。LBS 给 SNS 应用增加了地点的“纬度”，比如：你发布了一张照片标识在中关村，这样你的这个故事就会非常完整，好友们就会知道这个故事是在哪里发生的；你去了国贸，在人人网上你的好友发现你的标识是嘉里中心，那就可以在一起吃个饭。所以在移动互联网时代 SNS 有了 LBS 以后，人与人之间的沟通就变得更加真实，我们不过了解到你在做什么，还知道你哪里，极大增加了社交关系。”

吴疆认为 SNS 新的盈利模式也会是建立在 SoLoMo 的基础上。他以人人网为例，人人网旗下的团购网站“糯米网”并不是完全地、简单地依托人人网的用户资源，有效地利用用户的位置地点，提供更加贴近用户的服务，比如“你用人人网的移动客户端，到了某一地区，人人网就会告诉附近有哪些糯米的团购信息。你就可以在手机上实施这种购买。”这正是 SNS 网站向移动商务方向走出坚实的一步！

SNS 是移动互联网生态系统中重要一环

如何使 SNS 成为移动互联网中重要的一环，吴疆反复提到了一个词：“合作”。吴疆相信 SNS 会成为一个跨平台、跨设备的通讯平台。

总结

随着社会的发展，SNS 与移动互联网的交集越来越大。如何做好移动互联网的 SNS，并利用 SNS 网站海量的用户群体扩大应用使用范围，这是一道摆在吴疆和人人网的移动团队面前的算术题。尽管还没有答案、尽管前行的路还很崎岖，但是吴疆和他团队打造的移动互联网生态圈，还是让我们对人人网充满了期待。

完整文章请查看原文：

http://mobile.51cto.com/AppFocus-271559.htm ■